UNIVERSITY COLLEGE LONDON

MSC. MACHINE LEARNING

Multi-Target Tracking of Interacting Targets in the Presence of Occlusion

---

Author:

Sandeep BASAK

Supervisor:

Dr. Simon JULIER

This report is submitted as part requirement for the MSc Degree in Machine Learning at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

August, 2016

UNIVERSITY COLLEGE LONDON

Abstract

Multi-Target Tracking of Interacting Targets in the Presence of Occlusion

The aim of this project is to investigate methods to enhance the task of multi-object tracking of targets (vehicles) in the presence of occlusion. Furthermore such targets may *interact* with each other and as such their motion is not independent. The above cases frequently occur in real life situations for vehicles driving in dense urban traffic conditions where target vehicles may get occluded from time to time, increasing the difficulty of tracking them continuously. Moreover, vehicles on the road do not move completely independently and the presence/absence of other vehicles, their proximity, speed, direction of motion etc. all affect the motion dynamics of a particular target vehicle. This project aims to take such dependencies into account while tracking multiple *interacting* targets in the presence of prolonged periods of occlusion.

Acknowledgements

First, I want to thank my supervisor Dr. Simon Julier for introducing me to the tracking problem tackled in this project. I am also very grateful to him for his advice and plentiful suggestions during the course of this project as well as reviewing earlier drafts of this report. I also want to thank my parents and specially my partner Srijita for being a such a strong pillar of support all these past years. Credit for whatever I achieved goes to them, the mistakes are only mine.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Target tracking is a well established research area in the fields of engineering/robotics/computer vision. Broadly speaking, the problem is to establish the location (or other characteristics e.g. velocity/acceleration) of a target object as it evolves over time, given a series of noisy measurements obtained from sensors. Real life examples are aircraft tracking using radars, tracking vehicles moving over a road network [31], robot localisation [8], tracking a particular object through a video sequence, e.g. tracking of facial features [16], gait recognition [20] and people tracking [26]. Many of these problems in tracking consider the situation in which the targets of interest which are being tracked, occupy the same space. These include vehicle tracking and people tracking. Since these targets occupy the same space (say vehicles on a given road), the motion of such targets cannot be considered as independent and are often constrained by the presence of other surrounding targets and their motion dynamics. Most traditional models of tracking have ignored these inter-target dependencies. However, in this project we explicitly take such dependencies into account.

In our particular case, we are interested in investigating ways to track multiple vehicles in an dense traffic urban environment. This particular scenario presents its own unique set of challenges. The urban environment consists of streets, buildings and other vehicles. Often the target vehicle(s) may get hidden from the sensor field of view by say another vehicle or building. Thus there can be frequent and prolonged occlusions, where we do not receive any measurements at all from the sensors. There can also be shadows cast by buildings, which may throw off a tracker looking to track a vehicle based on its colour or other image characteristics (say shape or contour). This is because in a shadow region, the image quality deteriorates and as a result, the signal to noise ratio falls dramatically [5].

Furthermore, vehicles can also *disappear* by driving into car parks or other similar structures, or new vehicles may *appear* (i.e. join in) the group of vehicles we are tracking, causing problems for the tracker. Such appearance/ disappearance events may happen in occlusion zones (say vehicles driving into and out of a car park and the entrance/exit of the car park is hidden from the sensor field of view). Thus when the vehicles re-appear, (i.e. the sensor measurements resume), the tracker may find it difficult to re-capture the target object(s) (if they still exist) or to figure out which among the original set of vehicles that it was tracking, still exists. In addition, in a dense traffic urban environment, when vehicles are driving very close to each other, there can be additional complexities in data association, i.e. figuring out which sensor measurement belongs to which vehicle. Finally, as the sensor and vehicle detector are not perfect, there can be missed detections or false detections (i.e. clutter), creating additional complexities for the tracker .

Since we are tracking vehicles on a road, the motion dynamics of individual

vehicles are affected by the presence of other vehicles around them and their motion characteristics. Thus our target vehicles *interact* with each other as well as other non-target vehicles. We aim to capture this interaction in our work.

1.2 Objectives

The principal objective of this project is to investigate and develop multi-target tracking algorithms to track multiple *interacting* targets (vehicles in our case) in partially observable urban environments with prolonged periods of occlusion. By interacting targets, we mean that targets do not move independently of each other, i.e. their motion is dependent on one another. This corresponds to situations of dense urban traffic in which a target vehicle is always constrained by the vehicles surrounding it and cannot run over the surrounding vehicles.

Another objective of this project is to make inferences about the existence of target(s) once they emerge from the occlusion zone as well as make predictions of the state of such targets while they are in the occlusion zones.

The final goal is to test our methods on simulated datasets as a means of evaluating their performance.

1.3 Public Repositories

The code to implement all of the work described in this project can be found on the github account *https://github.com/quarkx33/Multi-Target-Tracking-Occlusion*.

1.4 Structure of the Report

In the next chapter we present a literature review of related past work done on this area. We argue that although the past research sheds light on various sub-parts of the problem we are interested in, none of it explicitly meets our project goals. In Chapter 3 we give details on how we build a simulator to model vehicle motion dynamics where the vehicles interact with each other. The outputs from this simulator would serve as the ground truth. Chapter 4 introduces the concept of a particle filter which is the method employed in this project to track the vehicles. This chapter also lays out the implementation details of the filter and its performance on our simulated scenarios. Chapter 5 adds the first layer of complexity to our problem where we extend our tracking method to deal with multi-target tracking of interacting vehicles *in the presence of occlusion zones*.

In Chapter 6 we add the second layer of complexity to our problem by allowing *new* vehicles to appear interspersed within our target group or *existing* vehicles in our target group to disappear *inside* the occlusion zones. We thus extend our tracking methodology to deal with this more complicated scenario. The tracker is designed to dynamically detect the occurrence of an event and adapt accordingly so as not to lose track of the original set of targets that it was tracking (as long as they exist). The chapter ends with a discussion of the results obtained by our method on the simulated scenarios. Finally, Chapter 7 concludes with thoughts on future directions for research in this area.

Chapter 2

Literature Review

2.1 Tracking Vehicles on Road Networks

The problem of tracking vehicles on road networks is a special case of a multi-target tracking problem in which there are multiple, non-overlapping targets which operate in 2D space and are largely constrained to linear features (roads) in the environment [6]. A number of systems have been developed which try to address this problem. These include the LoFT [23] and Aeschliman's system [5]. In these systems an operator indicates a vehicle of interest in a frame, and the tracking system tracks the motion of the targets over subsequent frames.

Approaches to tracking like the above examples are based on computer vision techniques and tend to use relatively simple tracking models. These systems often use basic motion models which assume piecewise constant velocity, i.e. if left to predict on their own, they assume that the target vehicle moves at a constant speed. They also do not account for interactions between vehicles, focussing solely on tracking the target of interest without reference to the effect of surrounding vehicles on the target's motion. Thus, if the target vehicles pass through prolonged periods of occlusion, these simplifying assumptions break down and the trackers often lose their targets. Hence the

above solutions need to be generalised in two ways: track multiple targets *and* account for the interaction between the targets.

2.2 Multi-Target Tracking

Various approaches have been developed for tracking multiple objects simultaneously. In *object-based* approaches, the tracker aims to enumerate all the objects and estimate their state. Multiple Hypothesis Tracking (MHT) [25] and Mahler's approach based on random finite sets (RFSs)[21] are the main approaches in this area. The idea is that given a set of measurements, all possible association hypotheses are enumerated and the best one selected. The main drawback of this approach is that the computational complexity increases factorially as the number of targets increases. There have been many strategies proposed to reduce the computational burden. See for example: [14], [33], [28], [34].

The main alternative to the above approach of enumerating all possible association hypotheses that does not have the computational burden, is based on *density-based* approaches. In this approach one estimates the *average* number of targets which can be found within a spatial region. The Probability Hypothesis Density (PHD) filter [21] and multi-Bernoulli filter [3] are examples of such approaches. Although they do not attempt to label each target individually (and thus have considerable computational advantages), they assume that all targets move independently of one another and thus the interactions between targets are ignored.

2.3 Interacting Targets

Although many tracking systems generally assume the motion of their targets as independent, this is rarely the situation in practice. In our case the targets are vehicles which often drive in busy roads with other vehicles. These other vehicles often constrain the motion of any particular target vehicle (in addition to considering road constraints). Hence we need to consider multi-target tracking in the context of dependant vehicular motion, i.e. *interacting* targets.

Many traffic simulation systems use agent-based approaches, where each vehicle is viewed as an active agent which can detect and respond to the state of other agents which surround it. Such systems are explored in [9] and [12]. However, despite the maturity of these models, they have not been widely adopted into tracking systems. Thus most vehicle tracking systems designed are not able to properly account for such traffic constraints. Since the traffic simulation systems described above are quite complex and detailed, to keep things simple, we would like to use a simple vehicle dynamics model which is easy to understand and yet produces the desired interactions between vehicles. Hence our first goal in this project is to develop a simple model of dependent vehicle motion which would serve as the ground truth and would subsequently be used to test the performance of our methods.

2.4 Tracking Interacting Targets

The problem of tracking targets whose movements are not independently and identically distributed are often addressed by approaches for *group tracking*.

Here one uses the idea that a set of targets form a group, within which members can interact with one another [27]. However, research in this area is focused on the problem of detecting when targets enter or leave groups. However, in our case of urban traffic congestion, there is no active attempt by vehicles to form groups with coordinated activities. Hence, group tracking does not directly apply to our problem.

One example of interacting targets without groups is considered in Khan[37], which considers the problem of ant tracking. Ants can be physically close to one another, but rarely climb over one another. To encode this constraint, a Markov random Field (MRF) was used which penalised the likelihood of ants climbing over one another. A particle filter was used with MCMC sampling to keep track of the ants. The results were fairly successful. Other more complicated approaches have also been tried for interacting targets, using multiple level hierarchical particle filters [29] and *context-aware sampling*[32] with varying degrees of success. However, note that none of the above methods address the tracking problem in the presence of occlusion.

2.5 Multi-Target Tracking in the Presence of Occlusion

Most tracking algorithms approach the occlusion problem by assuming that the probability of detection is zero in the occlusion zones and continue to apply standard tracking algorithms. This approach is sufficient if the target is not visible for short periods of time, but has problems when long periods of occlusion occur. This is because such systems generally assign a state-dependent probability of survival, which is less than 1. Thus if a target is not observed

for a few time-steps, the probability of survival falls below a threshold and is deleted. The standard approach for tracking in the presence of occlusions is to use *tracklets* [17]. In areas where targets are visible, track segments are created. Targets are lost when they move into areas of occlusion. Hypothesis testing is used to join the tracklets into tracks which pass through the occlusion regions. However, these systems lose all information about the targets as long as they are not visible. Hence, one of the goals of this project is to also make predictions on the state of the targets while they are passing through the occlusion regions.

In this chapter, we have briefly reviewed the past related work done in this area. We have also argued that none of the above work directly addresses our problem. In the next chapter we lay out the details of a vehicle motion model simulator that we have built which allows for dependent vehicle motion (i.e. interacting vehicles). The outputs from this simulator would serve as the ground truth.

Chapter 3

Vehicle Motion Model Simulator

3.1 Introduction

In order for the vehicle motion models to look realistic and to avoid using a simple piecewise constant velocity assumption, we can resort to detailed traffic flow simulation techniques. In general, there are two approaches when simulating traffic flows:

1. *Macroscopic models* consider the overall speed, density and flow rate of the traffic using a large number of agents and link them through macroscopic relations. See [9] and [12].

2. *Microscopic models* model the behaviour of each driver/vehicle, depending on its state and the state of the vehicle ahead, and thus simulates the position and speed of each vehicle on the road

For our case, microscopic models are more relevant as we are interested in tracking the overall state of each vehicle. The two classical models in the realm of microscopic models are the Gazis-Herman-Rothery (GHR) model and the Intelligent Driver Model (IDM), both of which generate realistic driving behaviours. Descriptions of such microscopic models can be found in [11]. In

particular, we mention Jonathan Chemla's 2014 UCL Master's Thesis [7] which describes the underlying dynamics equation of the above two classical models in more detail.

However, for the purposes of this project, we wanted to use a simple motion model simulator, so that we can better understand the state dynamics as well as perform testing of our methods. More sophisticated motion models such as those described above can be further explored as part of future work to see how our methods perform in such more realistic situations.

3.2 Motion Model - Motivation

In this section we describe a motion dynamics model that is used to simulate vehicle dynamics in our case. For our simple vehicle motion simulator, we have assumed that vehicles move in a straight line which represents the road (1D motion as if they are moving in a given lane on the highway). This is not an overly restrictive assumption as one can generalise to non-straight line (2D) motion by assuming all paths can be broken up into small approximately linear segments and the methods described later can be extended to such cases. Thus for our simulator, to keep things simple, we also assume that vehicles move on the x-axis towards the direction of the positive x-axis. Hence, the location of a vehicle is completely specified by its X coordinate, and so two vehicles cannot occupy the same x-position at any given time point.

The core underlying idea that the motion model intends to capture is that the vehicles *DO NOT* move independently, rather they interact with each other. Since, one vehicle can not drive through another and they are all travelling on a fixed straight line in the same direction, this means we cannot have vehicles overtake each other as that would mean one vehicle running over another

to get ahead of it. We also assume that if a vehicle is travelling in a certain direction (in this case the positive x-axis), it does not change/reverse its direction (for the duration of the simulation). Thus when a vehicle approaches very close to another vehicle ahead of it on the road, it slows down (brakes) as it cannot run-over the leading vehicle. Hence, its motion becomes constrained and this is how two vehicles *interact* with each other and their joint state can no longer be considered independent. Obviously, when two vehicles are far part, then for all practical purposes there is no such *interaction*.

The other assumption we have made is to assume that vehicles can be represented as point objects. Again, this is a simplifying assumption and in future work this could be relaxed to represent vehicles as extended objects.

3.3 Vehicle Motion Dynamics Model

In this section we describe the theory underlying our simulator for vehicle motion dynamics. This simulator was written in MATLAB. Our motion dynamics model will consist of simulating a platoon of vehicles. The front or leading vehicle will have a given behaviour (as it moves under no constraints), while the vehicles behind it are constrained by those in front. We first give the details of simulation of the front/leading vehicle. Define:

$\mathbf{x}_t^{(i)} \rightarrow$ state vector of vehicle i at time t

$z_t^{(i)} \rightarrow$ measurement/observation of vehicle i at time t.

In our case we assume we can only observe the position of the vehicle (i.e. $p_t^{(i)}$)

Hence we can write:

$$\mathbf{x}_t^{(i)} = \begin{pmatrix} p_t^{(i)} \\ v_t^{(i)} \\ a_t^{(i)} \end{pmatrix}$$

where, $p_t^{(i)}$ is the position at time t (for vehicle i)

$v_t^{(i)}$ is the velocity at time t (for vehicle i)

$a_t^{(i)}$ is the acceleration (or deceleration if negative) at time t

We assume that any given vehicle tends to drive at a constant speed μ under no constraining factors. Assuming piecewise constant acceleration, our motion model is:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \alpha & 1 \end{bmatrix} \begin{bmatrix} p_{t+\Delta t}^{(i)} \\ v_{t+\Delta t}^{(i)} \\ a_{t+\Delta t}^{(i)} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & \theta \end{bmatrix} \begin{bmatrix} p_t^{(i)} \\ v_t^{(i)} \\ a_t^{(i)} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \epsilon_t \end{bmatrix} \tag{3.1}$$

where ϵ_t is assumed to be Gaussian noise with mean $\alpha\mu$ and variance σ^2 ($\epsilon_t \sim \mathcal{N}(\alpha\mu, \sigma^2)$). If we expand out the last row for acceleration in the above equation, we have:

$$\begin{aligned} a_{t+\Delta t}^{(i)} &= -\alpha v_{t+\Delta t}^{(i)} + \theta a_t^{(i)} + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(\alpha\mu, \sigma^2) \\ \implies a_{t+\Delta t}^{(i)} &= \alpha\left(\mu - v_{t+\Delta t}^{(i)}\right) + \theta a_t^{(i)} + \epsilon_t^*, \qquad \epsilon_t^* \sim \mathcal{N}(0, \sigma^2) \end{aligned} \tag{3.2}$$

In the above form, acceleration is like a first order autoregressive process (AR(1)) with autoregression coefficient θ. It also has a mean reversion term with α as the mean reversion speed. Choosing $\alpha = 0$, cancels out this mean reversion factor. The mean reversion factor ensures that if velocity is very high ($v_{t+\Delta t}^{(i)} \gg \mu$), acceleration ($a_{t+\Delta t}^{(i)}$) is negative, so the vehicle slows down and vice versa. The second term ($\theta a_t^{(i)}$) ensures that $a_{t+\Delta t}^{(i)}$ has an effect from the acceleration $a_t^{(i)}$ at the previous time point. This ensures that $a_t^{(i)}$ varies smoothly with time. We want to avoid very jerky motions to simulate as closely as possible real life vehicle dynamics. In section 3.6, we show plots of how the trajectories of the vehicles look like.

3.3.1 Additional Constraints

We impose additional constraints on the values of acceleration (in order to ensure realistic motion without huge jumps). So we restrict simulated acceleration such that $a^{(i)}_{t+\Delta t} \in [a_{min}, a_{max}], \forall i = 1, 2, \ldots$, where a_{min} and a_{max} are the minimum and maximum allowed accelerations respectively.

Finally, we do not want $v^{(i)}_t < 0$, for any t, i.e. a vehicle cannot go backwards (i.e. change direction for the duration of the simulation). This implies that:

$$v^{(i)}_{t+\Delta t} \geq 0 \implies v^{(i)}_t + (\Delta t)\, a^{(i)}_t \geq 0 \implies a^{(i)}_t \geq -\frac{v^{(i)}_t}{\Delta t} \quad \forall t, \forall i. \tag{3.3}$$

We also assume that velocity cannot be greater than v_{max}. Thus:

$$v^{(i)}_{t+\Delta t} \leq v_{max} \implies v^{(i)}_t + (\Delta t)\, a^{(i)}_t \leq v_{max} \implies a^{(i)}_t \leq \frac{\left(v_{max} - v^{(i)}_t\right)}{\Delta t} \quad \forall t, \forall i. \tag{3.4}$$

Thus final acceleration $a^{(i)}_{t+\Delta t}$ simulated using 3.2 above is then restricted to satisfy 3.3, 3.4 and the min. and max. allowed values of acceleration as follows:

$$\text{Set:} \quad a^{(i)}_{t+\Delta t} = \min\left(\max\left(a^{(i)}_{t+\Delta t}, a_{min}, -\frac{v^{(i)}_{t+\Delta t}}{\Delta t}\right), a_{max}, \frac{\left(v_{max} - v^{(i)}_{t+\Delta t}\right)}{\Delta t}\right) \tag{3.5}$$

3.4 Motion Dynamics for Trailing Vehicles

The previous section described how acceleration is simulated for the leading vehicle (call it vehicle 1). For vehicles trailing behind, we have to impose additional constraints to ensure *no crossing or running over* of vehicles as well as induce *interaction* between vehicles. To implement this, two additional modifications are made as follows:

3.4.1 Modification 1 (No Running Over)

For vehicles 2, 3, ... onwards (i.e. i$\geq$ 2), we simulate the state vector $x^{(i)}_{t+\Delta t}$ via 3.1 as before. However, in this case we need to ensure that $p^{(i)}_{t+\Delta t} \leq p^{(i-1)}_{t+\Delta t}$. Thus we have:

$$
\begin{aligned}
& p^{(i)}_{t+\Delta t} \leq p^{(i-1)}_{t+\Delta t} \\
\implies & p^{(i)}_t + (\Delta t)\, v^{(i)}_t + \frac{1}{2}(\Delta t)^2 a^{(i)}_t \leq p^{(i-1)}_t + (\Delta t)\, v^{(i-1)}_t + \frac{1}{2}(\Delta t)^2 a^{(i-1)}_t \\
\implies & a^{(i)}_t \leq \frac{2}{(\Delta t)^2}\left[\left(p^{(i-1)}_t - p^{(i)}_t\right) + (\Delta t)\left(v^{(i-1)}_t - v^{(i)}_t\right) + \frac{1}{2}(\Delta t)^2 a^{(i-1)}_t\right] \\
\implies & a^{(i)}_t \leq c^{(i)}_t \quad \text{(say)} \qquad (3.6) \\
\text{where } c^{(i)}_t = & \frac{2}{(\Delta t)^2}\left[\left(p^{(i-1)}_t - p^{(i)}_t\right) + (\Delta t)\left(v^{(i-1)}_t - v^{(i)}_t\right) + \frac{1}{2}(\Delta t)^2 a^{(i-1)}_t\right]
\end{aligned}
$$

This additional constraint in 3.6 also needs to be taken into account for the trailing vehicles 2, 3, ... onwards (i.e. i$\geq$ 2).

3.4.2 Modification 2 (Speeding Up/Braking)

We define a safe distance s between vehicles. If the vehicle under consideration is behind the vehicle ahead by a distance *more* than the safe distance, then it speeds up, trying to reduce the gap. However, if the vehicle ahead of it is *less* than the safe distance, then it applies braking in order to attempt to increase the distance between them. The amount of speeding-up (acceleration) or braking (deceleration) is proportional to how much the vehicle is over or under the safe distance s respectively.

For vehicles 2, 3, ... onwards (i.e. i$\geq$ 2), we simulate the acceleration (say) $\tilde{a}^{(i)}_{t+\Delta t}$ via 3.2 as before, restricted to $[a_{min}, a_{max}]$.

The speeding up is modelled as follows:

Say $d^{(i)}_{t+\Delta t}$ = separation distance to vehicle ahead at time $(t+\Delta t) = p^{(i-1)}_{t+\Delta t} - p^{(i)}_{t+\Delta t}$.

Define $\gamma^{(i)}_{t+\Delta t} = \min\left(\max\left(e^{\left(1-\frac{s}{d^{(i)}_{t+\Delta t}}\right)} - 1, 0\right), 1\right)$, where s = safe distance.

(Note: $\gamma^{(i)}_{t+\Delta t} \in [0,1]$ and $d^{(i)}_{t+\Delta t} \leq s \implies \gamma^{(i)}_{t+\Delta t} = 0$ and $\gamma^{(i)}_{t+\Delta t}$ increases as $\left(d^{(i)}_{t+\Delta t} - s\right)$ increases). Thus the above form of $\gamma^{(i)}_{t+\Delta t}$ ensures that as the separation distance $d^{(i)}_{t+\Delta t}$ between vehicles increases, the $\gamma^{(i)}_{t+\Delta t}$ term approaches 1.

So we set $a^{(i)}_{t+\Delta t} = \gamma^{(i)}_{t+\Delta t} a_{max} + (1 - \gamma^{(i)}_{t+\Delta t})\tilde{a}^{(i)}_{t+\Delta t}$ (weighted combination of simulated acceleration and max. allowed acceleration). Thus if the separation distance between the vehicles increases to more than the pre-specified safe distance, the above term ensures that a speeding up element kicks in (which is directly proportional to the separation distance), thus speeding up the trailing vehicle.

The braking term is modelled as follows: Define $\beta^{(i)}_{t+\Delta t} = \max\left(0, 1 - \frac{d^{(i)}_{t+\Delta t}}{s}\right)$, where s = safe distance. (Note: $\beta_{t+\Delta t} \in [0,1]$).

We set $a^{(i)}_{t+\Delta t} = \beta^{(i)}_{t+\Delta t} a_{min} + (1 - \beta^{(i)}_{t+\Delta t})\tilde{a}^{(i)}_{t+\Delta t}$ (weighted combination of simulated acceleration and max. allowed deceleration). Thus if the separation distance between the vehicles falls below the pre-specified safe distance, the above term ensures that a braking element kicks in (which is inversely proportional to the separation distance), thus slowing down the trailing vehicle.

In a congested urban traffic scenario, where vehicles are driving close to each other, if a substantial gap opens up between two vehicles, then the trailing vehicle often speeds up to close/reduce the gap. However once it gets too close to the vehicle ahead, it starts to slow down to maintain a reasonable distance to the vehicle ahead and avoid bumping into it. If the gap between them again starts to increase too much, then the trailing vehicle has to start speeding up

again. It is this alternate speeding up/braking effects that the speeding up and braking terms introduced above intends to capture.

The braking term in particular helps to introduce *interactions* between vehicles by influencing the motion of a particular vehicle given the vehicle ahead of it. As is expected, when the vehicles are far apart, this term has no effect. The speeding up term on the other hand, helps to ensure that vehicles often get close to each other and thus *interact*.

Finally, we restrict $a^{(i)}_{t+\Delta t}$ as simulated above to satisfy 3.3, 3.4, 3.6 and the min. and max. allowed values of acceleration as follows. Set:

$$a^{(i)}_{t+\Delta t} = \min\left(\max\left(a^{(i)}_{t+\Delta t}, a_{min}, -\frac{v^{(i)}_{t+\Delta t}}{\Delta t}\right), a_{max}, \frac{\left(v_{max} - v^{(i)}_{t+\Delta t}\right)}{\Delta t}, c^{(i)}_{t+\Delta t}\right) \quad (3.7)$$

However it is possible that the upper bound in 3.6 is less than the lower bounds of a_{min} and $-\frac{v^{(i)}_{t+\Delta t}}{\Delta t}$ (refer 3.3). In this case no valid value of acceleration exists that would satisfy 3.7 above. When this is the case, we set acceleration to its minimum possible value (equal to $-\frac{v^{(i)}_{t+\Delta t}}{\Delta t}$ as given by 3.3 assuming that we have say *emergency* braking).

Finally, we also need to check that we have $p^{(i)}_{t+\Delta t} \leq p^{(i-1)}_{t+\Delta t} \quad \forall i \in \{2, 3, \ldots\}$. If this is violated for any i, then we forcibly ensure this by generating: $p^{(i)}_{t+\Delta t} = p^{(i)}_t + \mathcal{U}\left(0, p^{(i-1)}_{t+\Delta t} - p^{(i)}_t\right)$, where $\mathcal{U}(a, b)$ denotes the uniform distribution over $[a, b]$. This correction ensures that vehicle i never overtakes the vehicle $(i-1)$ before it. This particular adjustment can be viewed as a weakness of the simulator, where we break our earlier assumption of piecewise constant acceleration. However on empirical investigation, we found that this condition occurred rarely in practice. As mentioned in the introduction of this chapter, in future, more sophisticated motion models (without such weaknesses) may

be considered as further extensions of this work.

3.5 Values Chosen for the Parameters of the Motion Model

For our goal to simulate vehicular dynamics in an dense traffic urban environment (where vehicles are closely spaced to each other), we chose the following parameters:

$a_{max} = 1$ (in m/s^2),

$a_{min} = -2$ (in m/s^2),

$v_{max} = 10$ (in m/s),

$\mu = 5$ (mean velocity in m/s),

$\alpha = 0.08$ (mean reversion speed),

$\theta = 0.75$ (autoregressive term in acceleration equation),

$\sigma = 0.09$ (noise term in simulation of acceleration 3.2),

$s = 8$ (safe distance in meters to determine braking/speeding up)

$\Delta t = 0.1$ (we are modelling motion dynamics per $\frac{1}{10}$th of a second)

The above values were fine-tuned by hand and were experimentally tested by running the simulator multiple times to see if it was producing the desired characteristics (i.e. vehicle interactions) as would be expected in an urban scenario. We show below in Figure 3.1 sample plots of the trajectories of 3 vehicles (with and without vehicle interaction).

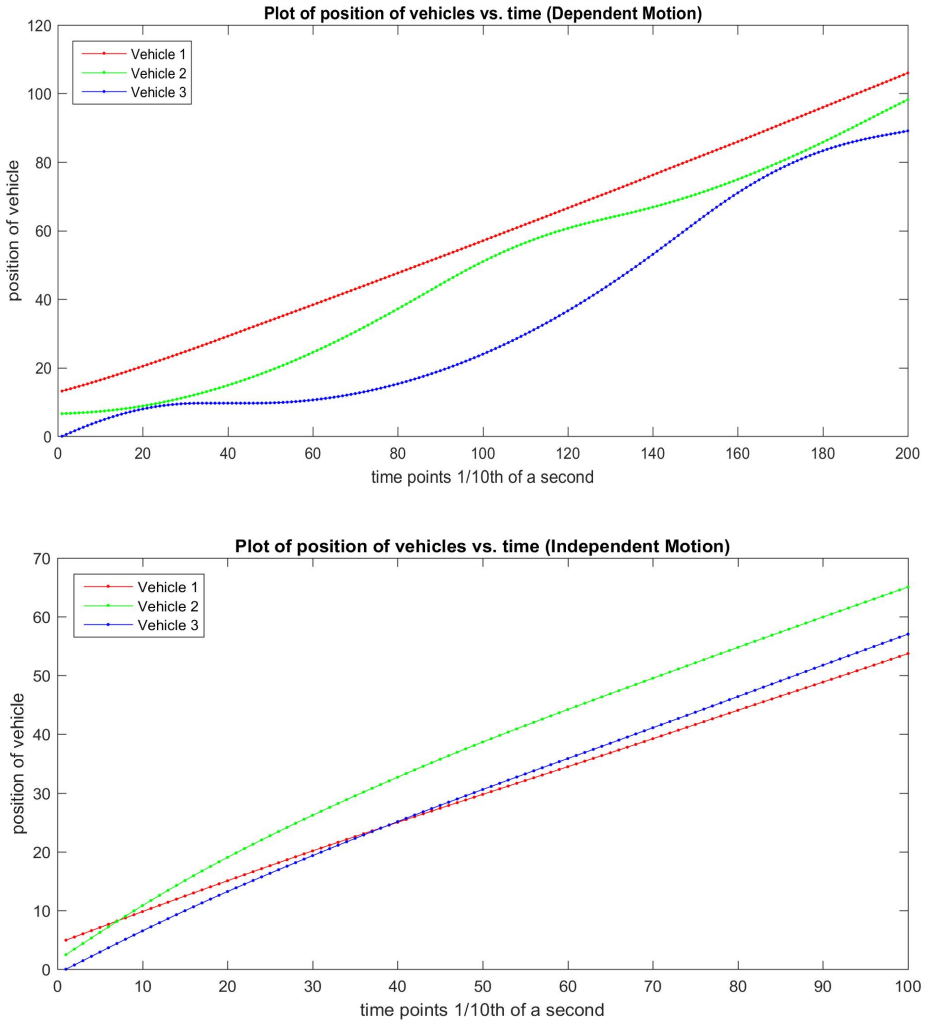

FIGURE 3.1: Trajectories of Dependent vs Independent Vehicle Motion

As can be seen above, in the dependent case (top figure), the vehicles never overtake/run-over each other. When one vehicle starts to approach very near to the vehicle ahead, it slows down and subsequently the gap between them increases. However, in the bottom figure, where we do not have *interactions* between vehicles (i.e. the vehicles move independently), we see that *crossing over* happens and although vehicle 1 starts ahead of the other vehicles, it ends up trailing all of them. After some time, all vehicles (in the independent case)

attain the long term average speed of 5m/s and hence the separation distance between them remains almost constant. Since we have restricted the motion of the vehicles to be only on the x-axis, hence the only way that vehicles can *overtake* one another is if they run over each other. This is not reflective of real life scenarios and hence shows why we need to model dependent vehicle motion.

3.6 Illustrative Plots of Simulated Vehicle Motion

We show below sample plots of trajectories obtained from the simulator. The plot below shows the trajectories of two vehicles (which also shows how the separation distance between them oscillates).

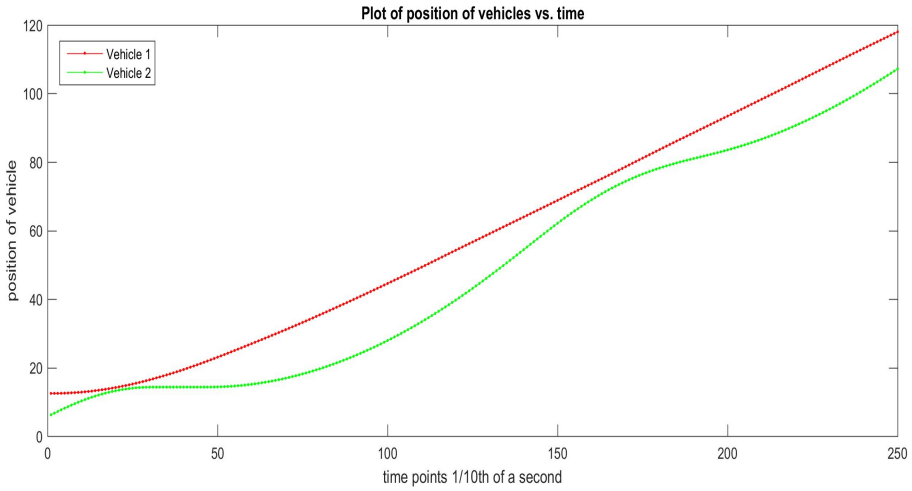

FIGURE 3.2: Plot showing how the separation distance oscillates between the Leading and the Trailing vehicle

The plot above shows that as the trailing vehicle approaches the vehicle ahead, the separation distance falls below the safe distance, causing braking. Similarly when the separation distance goes over the safe distance, the trailing vehicle speeds up. This behaviour should generate oscillatory effects in the

velocity and acceleration of the trailing vehicle. We see this in the following plots.

The plots below are sample plots of how the velocity varies for the leading vehicle (i.e. independent motion) and the vehicle trailing it (i.e. dependent motion).

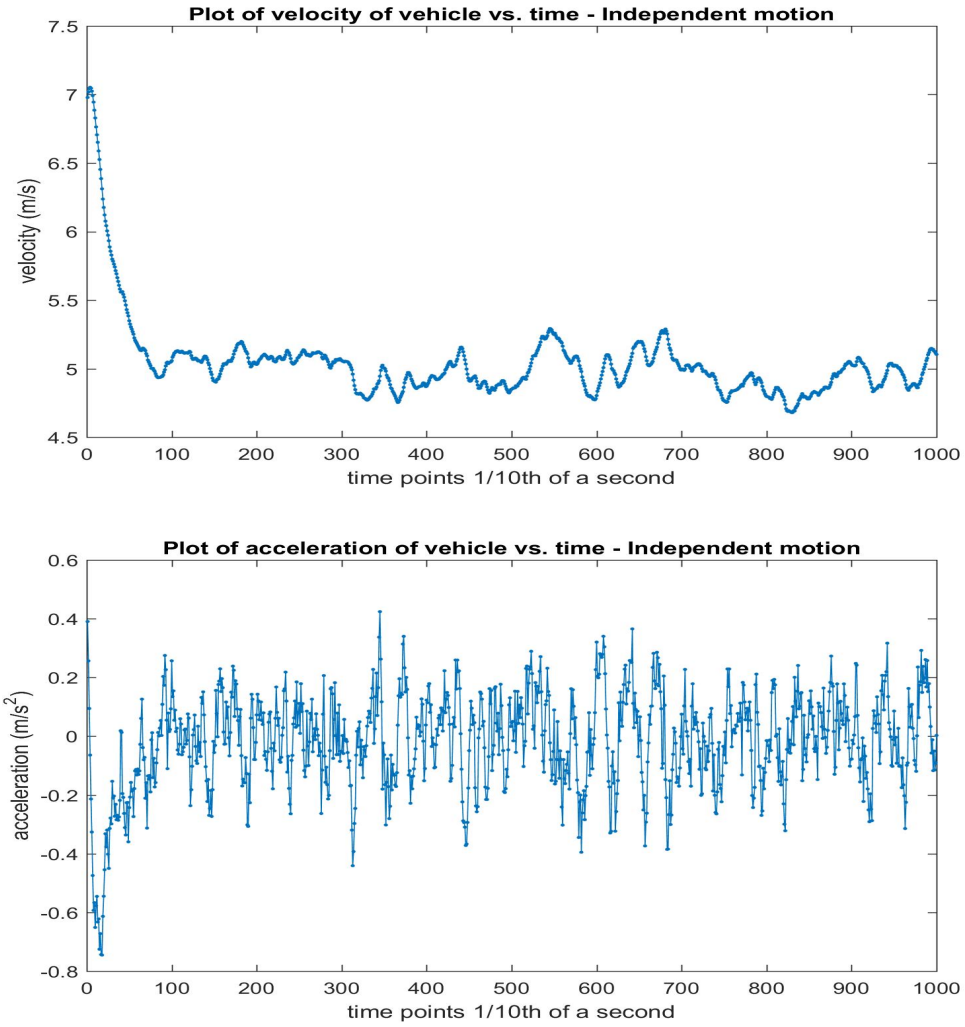

FIGURE 3.3: Velocity and acceleration for vehicle 1 (i.e. Independent motion)

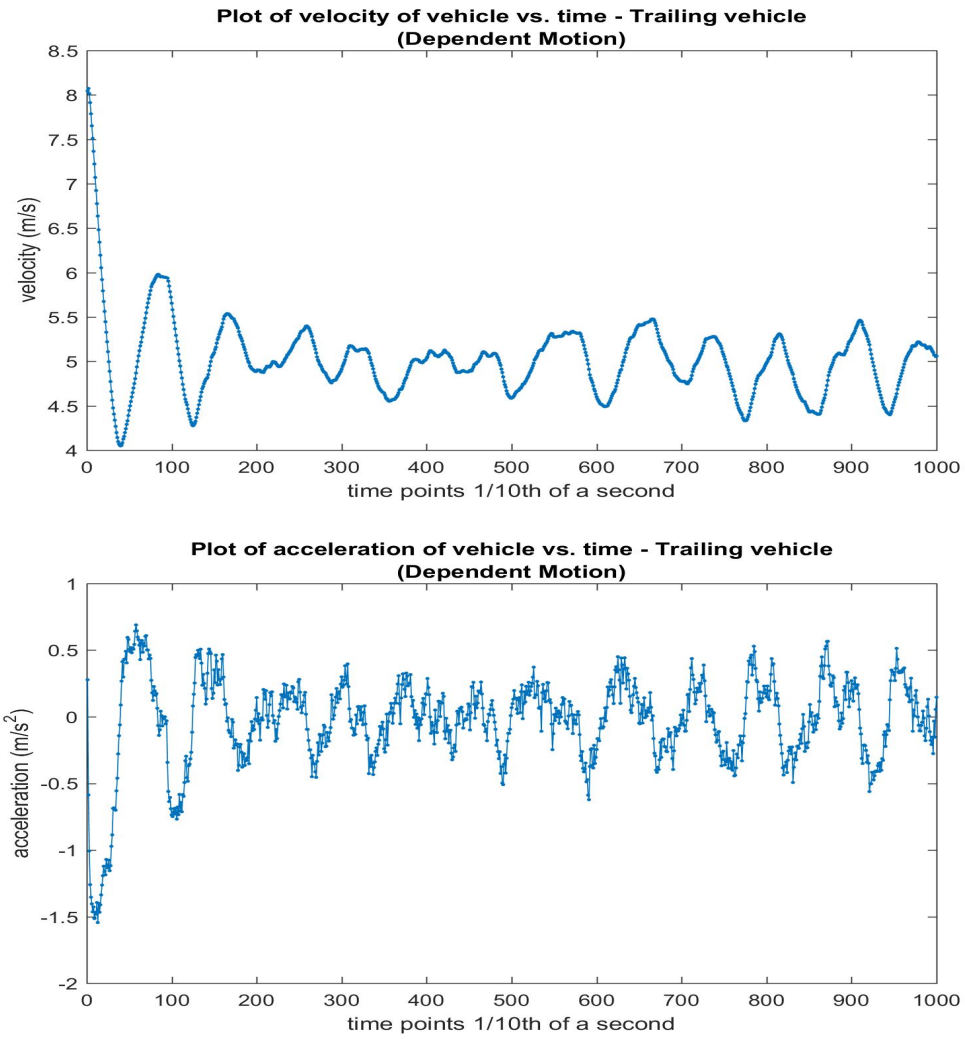

FIGURE 3.4: Velocity and acceleration for vehicle 2 (i.e. motion with *interaction* with vehicle 1)

As the figures above show, for vehicle 1 which drives under no constraints, the velocity slowly converges to the long term mean of 5m/s and the acceleration oscillates around 0 (Figure 3.3). However, for the vehicle trailing behind it, we see that the velocity increases and decreases in an oscillatory fashion, and so does the acceleration (Figure 3.4). This oscillatory motion is due to the speeding and braking terms introduced earlier. Note that the values of velocity and acceleration are bounded by their respective upper and lower limits.

In this chapter, we have given the details of our vehicle motion model simulator which would serve as the ground truth. In the next chapter, we describe in detail our tracking method used to track the vehicles being modelled in the simulator.

Chapter 4

Using a Particle Filter to Track Vehicles

4.1 Introduction

In this chapter we first introduce the particle filter which forms the backbone of the recursive estimation procedure employed in this project to track the vehicles. Next we lay out the implementation details of the particle filter in our simulated setup and then show the results obtained. The objective here is to simulate vehicle dynamics using our simulator and then use a particle filter to track those vehicles using inputs from a noisy sensor.

There are a huge range of applications that require on-line estimates and predictions of an evolving set of parameters given uncertain data and dynamics — e.g. object tracking, forecasting of financial indices, vehicle navigation and control, numerical weather prediction etc. An obvious common framework for addressing such problems would consist of a motion *dynamics model* (describing the evolution of the system over time; also called a *temporal model*) and a *measurement model* that describes how the available data is related to the system.

When expressing such models in a probabilistic form, a Bayesian approach to estimation of the state vector aims to construct the posterior probability density (pdf) of the target's state vector using all available information (i.e. measurements) up until that time point. The recursive Bayesian filter provides a formal mechanism for propagating and updating the posterior pdf as new information (measurements) are received over time.

A particle filter is an implementation of the recursive Bayesian filter using (sequential) Monte Carlo methods. Instead of describing the required pdf as a functional form, in a particle filter; the pdf is represented approximately as a set of random samples drawn from the pdf. The approximation may be made as accurate as possible by increasing the number of samples. For multivariate pdfs, the samples are vectors. It is these random samples that are called *particles* which are propagated and updated according to the dynamics of the temporal and measurement models. The principal attraction of this method is that we are no longer restricted to distributions with closed form analytic expressions. Thus this method greatly expands the range of problems that can be tackled while still remaining within the Bayesian framework.

4.2 Underlying Theory of the Particle Filter

The aim of the particle filter is to estimate the pdf associated with the state vector of the target of interest. This estimation problem assumes two fundamental mathematical models: the state dynamics model (or temporal model) and the measurement model. The dynamics model describes how the state vector evolves with time and is assumed to be of the form

$$\mathbf{x}_{t+\Delta t} = \mathbf{f}_t\left(\mathbf{x}_t, \mathbf{e}_t\right), \quad t > 0 \tag{4.1}$$

Here $\mathbf{x}_{t+\Delta t}$ is the state vector to be estimated, Δt denotes the time step and $\mathbf{f}_t$ is a known possibly non-linear function. $\mathbf{e}_t \ \forall t$ are independent and identically distributed noise sequence, whose pdf is assumed known. Note that 4.1 defines a first order Markov process and an equivalent probabilistic notation of the state evolution is $p\left(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t\right)$, which is sometimes called the transition density. For the special case when $\mathbf{f}$ is linear and $\mathbf{e}_t$ is Gaussian, the transition density $p\left(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t\right)$ is also Gaussian.

The measurement equation relates the received sensor measurements to the state vector:

$$\mathbf{z}_t = \mathbf{h}_t\left(\mathbf{x}_t, \mathbf{w}_t\right), \quad t > 0 \tag{4.2}$$

where $\mathbf{z}_t$ is the vector of received measurements at time step t, $\mathbf{h}_t$ is the known measurement function and $\mathbf{w}_t \ \forall t$ form a white noise sequence (the measurement noise or error). Again, the pdf of $\mathbf{w}_t$ is assumed known and $\mathbf{e}_t$ and $\mathbf{w}_t$ are mutually independent. Thus an equivalent probabilistic description for 4.2 is the conditional pdf $p\left(\mathbf{z}_t|\mathbf{x}_t\right)$. For the special case when $\mathbf{h}_t$ is linear and $\mathbf{w}_t$ is Gaussian, $p\left(\mathbf{z}_t|\mathbf{x}_t\right)$ is also Gaussian.

The final piece of information to complete the specification of the estimation problem are the initial conditions. This is the prior pdf $p\left(\mathbf{x}_0\right)$ of the state vector at time $t = 0$, before any measurements have been received. So in summary, the probabilistic description of the problem is: $p\left(\mathbf{x}_0\right)$, $p\left(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t\right)$ and $p\left(\mathbf{z}_t|\mathbf{x}_t\right)$.

4.2.1 Bayesian Filter

In the Bayesian approach, we attempt to construct the posterior pdf of the state vector $\mathbf{x}_{t+\Delta t}$ given all the available information. This posterior pdf at time step

$t + \Delta t$ is written as $p\left(\mathbf{x}_{t+\Delta t}|\mathbf{Z}_{t+\Delta t}\right)$, where $\mathbf{Z}_{t+\Delta t}$ denotes the set of all measurements received upto and including $\mathbf{z}_{t+\Delta t}$: $\mathbf{Z}_{t+\Delta t} = \{\mathbf{z}_{t+\Delta t}, \mathbf{z}_t, \mathbf{z}_{t-\Delta t}, \mathbf{z}_{t-2\Delta t}, \ldots\}$

The Bayesian recursive filter consists of a prediction and an update operation. The prediction operation propagates the posterior pdf of the state vector from time step t forwards to time step $t + \Delta t$. Suppose that $p\left(\mathbf{x}_t|\mathbf{Z}_t\right)$ is available, then $p\left(\mathbf{x}_{t+\Delta t}|\mathbf{Z}_t\right)$, the prior pdf of the state vector at time step $t + \Delta t$ may be obtained via the transition density:

$$\underbrace{p\left(\mathbf{x}_{t+\Delta t}|\mathbf{Z}_t\right)}_{\text{Prior at } t+\Delta t} = \int \underbrace{p\left(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t\right)}_{\text{transition density}} \quad \underbrace{p\left(\mathbf{x}_t|\mathbf{Z}_t\right)}_{\text{Posterior from } t} \; d\mathbf{x}_t. \tag{4.3}$$

This is known as the Chapman-Kolmogorov equation.

The prior pdf may be updated to incorporate the new measurements $\mathbf{z}_{t+\Delta t}$ to give the required posterior pdf at time step $t + \Delta t$:

$$\underbrace{p\left(\mathbf{x}_{t+\Delta t}|\mathbf{Z}_{t+\Delta t}\right)}_{\text{Posterior}} = \underbrace{p\left(\mathbf{z}_{t+\Delta t}|\mathbf{x}_{t+\Delta t}\right)}_{\text{Likelihood}} \underbrace{p\left(\mathbf{x}_{t+\Delta t}|\mathbf{Z}_t\right)}_{\text{Prior}} \quad / \quad \underbrace{p\left(\mathbf{z}_{t+\Delta t}|\mathbf{Z}_t\right)}_{\substack{\text{Normalising} \\ \text{Factor}}} \tag{4.4}$$

This is Bayes rule, where the normalising denominator is given by $p\left(\mathbf{z}_{t+\Delta t}|\mathbf{Z}_t\right)$. Equations 4.3 and 4.4 define the Bayesian recursive filter with initial condition given by the specified prior pdf $p\left(\mathbf{x}_0|\mathbf{Z}_0\right) = p\left(\mathbf{x}_0\right)$ (where $\mathbf{Z}_0$ is interpreted as the empty set).

Only in special cases, can one combine equations 4.3 and 4.4 to obtain a closed form expression for the posterior. The most important of these special cases is the Linear Gaussian model: if $p\left(\mathbf{x}_0\right)$, $p\left(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t\right)$ and $p\left(\mathbf{z}_t|\mathbf{x}_t\right)$ are all Gaussian, then the posterior density remains Gaussian ([13]) and 4.3 and 4.4 reduce to the standard Kalman filter (which recursively specifies the mean and covariance of the posterior Gaussian).

However, for cases in which the transition density $p(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t)$ or the measurement model $p(\mathbf{z}_t|\mathbf{x}_t)$ is not Gaussian, the Kalman filter (or extensions of it) do not perform very well. This is because these filters represent the uncertainty over the states as a Gaussian distribution. They are hence ill-equipped to deal with situations where the true probability distribution over the states is say, multi-modal. A Particle filter resolves this problem by representing the probability density as a set of particles in the state space while making *no* distributional assumptions. This property allows it to deal effectively with any distributions (including multi-modal distributions). The following section describes this filter in more detail.

4.2.2 Particle Filter Algorithm

Suppose that a set of N random samples from the posterior pdf $p(\mathbf{x}_t|\mathbf{Z}_t)$ are available. We denote these samples or particles by $\{\mathbf{x}_t^{i*}\}_{i=1}^N$.

The **prediction** phase of the filter consists of passing each of these samples from time step t through the dynamics model 4.1 to generate a set of prior samples at time step t. These prior samples are written as $\{\mathbf{x}_{t+\Delta t}^i\}_{i=1}^N$, where

$$\mathbf{x}_{t+\Delta t}^i = \mathbf{f}_t\left(\mathbf{x}_t^{i*}, \mathbf{e}_t^i\right), \quad \forall i = 1, \ldots, N$$

and e_t^i are (independent) samples drawn from the pdf of the system noise. This step generates a set of samples or particles from the prior pdf $p(\mathbf{x}_{t+\Delta t}|\mathbf{Z}_t)$. This step is often called the **temporal step**. To **update** the particles in light of the measurement $\mathbf{z}_{t+\Delta t}$, a weight $\tilde{w}_{t+\Delta t}^i$ is calculated for each particle. This weight is the measurement likelihood evaluated at the value of the prior sample: $\tilde{w}_{t+\Delta t}^i = p\left(\mathbf{z}_{t+\Delta t}|\mathbf{x}_{t+\Delta t}^i\right)$. The weights are then normalised so that they sum

to 1: $w^i_{t+\Delta t} = \tilde{w}^i_{t+\Delta t} / \sum_{j=1}^{N} \tilde{w}^j_{t+\Delta t}$ and the prior particles are resampled (with replacement) according to these normalised weights to produce a new set of particles:

$$\{\mathbf{x}^{i*}_{t+\Delta t}\}_{i=1}^{N} \quad \text{such that} \quad \mathbf{P}\left(\mathbf{x}^{i*}_{t+\Delta t} = \mathbf{x}^{j}_{t+\Delta t}\right) = w^i_{t+\Delta t}, \quad \forall i, \forall j \in [1, \ldots, N]$$

This update step is often called the **measurement update step**. We say that the new set of particles are samples from the required pdf $p\left(\mathbf{x}_{t+\Delta t}|\mathbf{Z}_{t+\Delta t}\right)$ and thus a cycle of the algorithm is complete. Note that the update procedure effectively weights each prior sample of the state vector by its plausibility with respect to the latest measurement. The figure below shows a depiction of the particle filter at work.

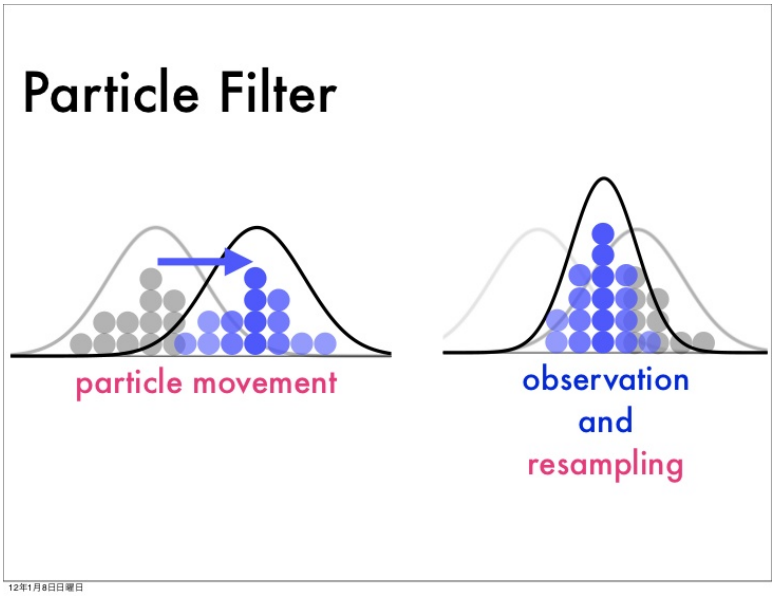

FIGURE 4.1: Particle Filter Visualisation

The above algorithm is often known as the Sampling Importance Resampling (SIR) filter and it was introduced in 1993 [22] where it was called the bootstrap filter. It was independently proposed by a number of other research groups including Kitagawa [19] as a Monte Carlo filter and Isard and Blake

[15] as the CONDENSATION algorithm. Since then, there have been many extensions and generalisations to the basic particle filter. For a broad discussion on particle filters and their various extensions, we refer the reader to [30]. For our project, we stick to the basic particle filter.

In conclusion, it must be mentioned that the main disadvantage of particle filters is their computational cost: in high dimensions, a very large number of particles may be required to get an accurate representation of the true distribution over the states. We discuss these issues in the context of our project in later sections.

4.3 Ground Truth Generation

In order to determine how well the particle filter performs in our setup, we need to first generate the ground truth by simulating vehicle motion over time using the simulator described in Chapter 3. In order to keep computational costs low in this stage, we have assumed that we are dealing with three vehicles (say vehicles 1, 2 and 3), where vehicle 1 is ahead of vehicle 2, which is ahead of vehicle 3. We randomly initialise the starting positions, velocity and acceleration of the vehicles such that at time $t = 1$, we have $p_1^{(1)} > p_1^{(2)} > p_1^{(3)}$ and $v_1^{(1)}, v_1^{(2)}, v_1^{(3)} \in [0, v_{max}]$ and $a_1^{(1)}, a_1^{(2)}, a_1^{(3)} \in [a_{min}, a_{max}]$. Given these initial starting positions, we then run the simulator for a given time frame (in our case we have assumed 100 seconds), and collect the state vectors, i.e. $\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \mathbf{x}_t^{(3)}\ \forall t \in 1, 2, \ldots, 100$. These state vectors will serve as the ground truth.

Next we simulate the measurement process, i.e collecting measurements of the vehicle positions using a noisy sensor as in 4.2. The measurements

are simulated by assuming that $p\left(z_t^{(i)}|\mathbf{x}_t^{(i)}\right) \sim \mathcal{N}(0, \sigma^2),\ \forall i \in \{1,2,3\}, \forall t \in \{1,2,\ldots,100\}$, where we assume $\sigma = 3$. Hence we are assuming that in our case the sensor is quite inaccurate, which is reflected by the high value of σ chosen above. Note that $z_t^{(i)}$s above are scalars, since the sensor is only collecting the positions of the vehicles. It is these sensor measurements that are fed into the particle filter which is used to track the state vector of the vehicles. We also assume that we receive one measurement every second from the sensor.

4.4 Particle Filter Performance

The particle filter used in this project was implemented in MATLAB. The key parameter to choose in the particle filter is the number of particles that we should use. We ran multiple experiments with various number of particles to determine an appropriate number. The two main statistics we looked at to determine filter performance are the MSE and the Mahalanobis distance. We explain the computation of these in detail below.

Given that we are modelling the position, velocity and acceleration of each vehicle, so in effect each particle is a 9-dimensional vector (3 dimensions for each of the 3 vehicles). We represent this vector by:

$$\mathbf{X}_{t,k} = \begin{pmatrix} \mathbf{x}_{t,k}^{(1)} \\ \mathbf{x}_{t,k}^{(2)} \\ \mathbf{x}_{t,k}^{(3)} \end{pmatrix}$$

where $\mathbf{x}_{t,k}^{(i)}\ \forall i \in \{1,2,3\}$ represents the state vector for the kth particle for vehicle i at time point t and is defined as in 3.3.

Now given an input number of particles (say N), for each time point t the particle filter generates N particles $\{\mathbf{X}_{t,k}\}_{k=1}^{N} \in \mathbb{R}^9$. Using the particles to represent independent samples from the posterior distribution of the state space of the vehicles, we can then estimate the mean position of each vehicle at time t by averaging over the position estimates from these N particles. Thus we define:

$\hat{p}_{t,k}^{(i)} \rightarrow$ position estimate from particle k at time t for vehicle i
$\hat{p}_{t}^{(i)} = \frac{1}{N}\sum_{k=1}^{N} \hat{p}_{t,k}^{(i)} \rightarrow$ mean position estimate of filter at time t for vehicle i
$E_t^{(i)} = \hat{p}_t^{(i)} - p_t^{(i)}$ = error in the filter estimate at time t for vehicle i.

$$\mathbf{E}_t = \begin{pmatrix} E_t^{(1)} \\ E_t^{(2)} \\ E_t^{(3)} \end{pmatrix}, \quad \bar{\mathbf{X}}_t = \frac{1}{N}\sum_{k=1}^{N} \mathbf{X}_{t,k}, \qquad \mathbf{\Sigma}_t = \frac{1}{N}\sum_{k=1}^{N} \left(\mathbf{X}_{t,k} - \bar{\mathbf{X}}_t\right)\left(\mathbf{X}_{t,k} - \bar{\mathbf{X}}_t\right)^T$$

In the above, $\mathbf{\Sigma}_t$ represents the covariance estimate of the particles (for the full state space). We then define the Mahalanobis distance as: $Mahal_t = {\mathbf{E}_t}^T \mathbf{\Sigma}_t^{-1} \mathbf{E}_t$.

Setting a fixed number of simulations, we calculate the position errors ($E_t^{(i)}$ as above for each vehicle separately) and the Mahalanobis distance at each time point ($Mahal_t$), for each simulation. We then compute the MSE of the position errors per vehicle; as well as the average Mahalanobis distance (for the full state vector) over all the simulations, separately for each time point. For clarity, if we define:

$E_{t,s}^{(i)} \rightarrow$ error in filter estimate at time t for vehicle i, simulation s
$\bar{E}^{(i)} = \frac{1}{ST}\sum_{s=1}^{N}\sum_{t=1}^{T} E_{t,s}^{(i)}$, then the MSE of the position errors for vehicle i are defined as: $MSE^{(i)} = \frac{1}{ST}\sum_{s=1}^{S}\sum_{t=1}^{T}\left(E_{t,s}^{(i)} - \bar{E}^{(i)}\right)^2$, where S = number of simulations, T = number of time points simulated.

Note that the MSE says, in absolute terms, how large is the error in the

position estimates from the particle filter for a given vehicle. For vehicles 1 and 3, the MSE in their positions should be upper bounded by the measurement error. For vehicle 2, it will be a function of the distance between vehicles 1 and 3.

The Mahalanobis distance on the other hand determines if the error is commensurate with how big we think the error should be. This quantity should converge to a value about equal to the dimension of the state space (i.e. 9 in our case).

4.5 Results and Conclusions

We ran the above particle filter for 5,000 simulations (each simulation consisting of 100 seconds). The results are summarised below:

Number of particles	**1,000**	**5,000**	**10,000**
MSE: All vehicle positions	2.3823	2.2938	2.3048
MSE: vehicle 1 position	0.5209	0.5104	0.5182
MSE: vehicle 2 position	0.6787	0.6587	0.6626
MSE: vehicle 3 position	1.1828	1.1247	1.1239
Mahalanobis distance (full state space)	10.2803	9.0823	9.0864

TABLE 4.1: Summary of Particle filter statistics averaged over 5,000 simulations

As can be seen above, we get good convergence at 1,000 particles and really the only variation in the above table is due to the variance in the simulations. However, if we plot the Mahalanobis distance for individual time points, we get the following plots:

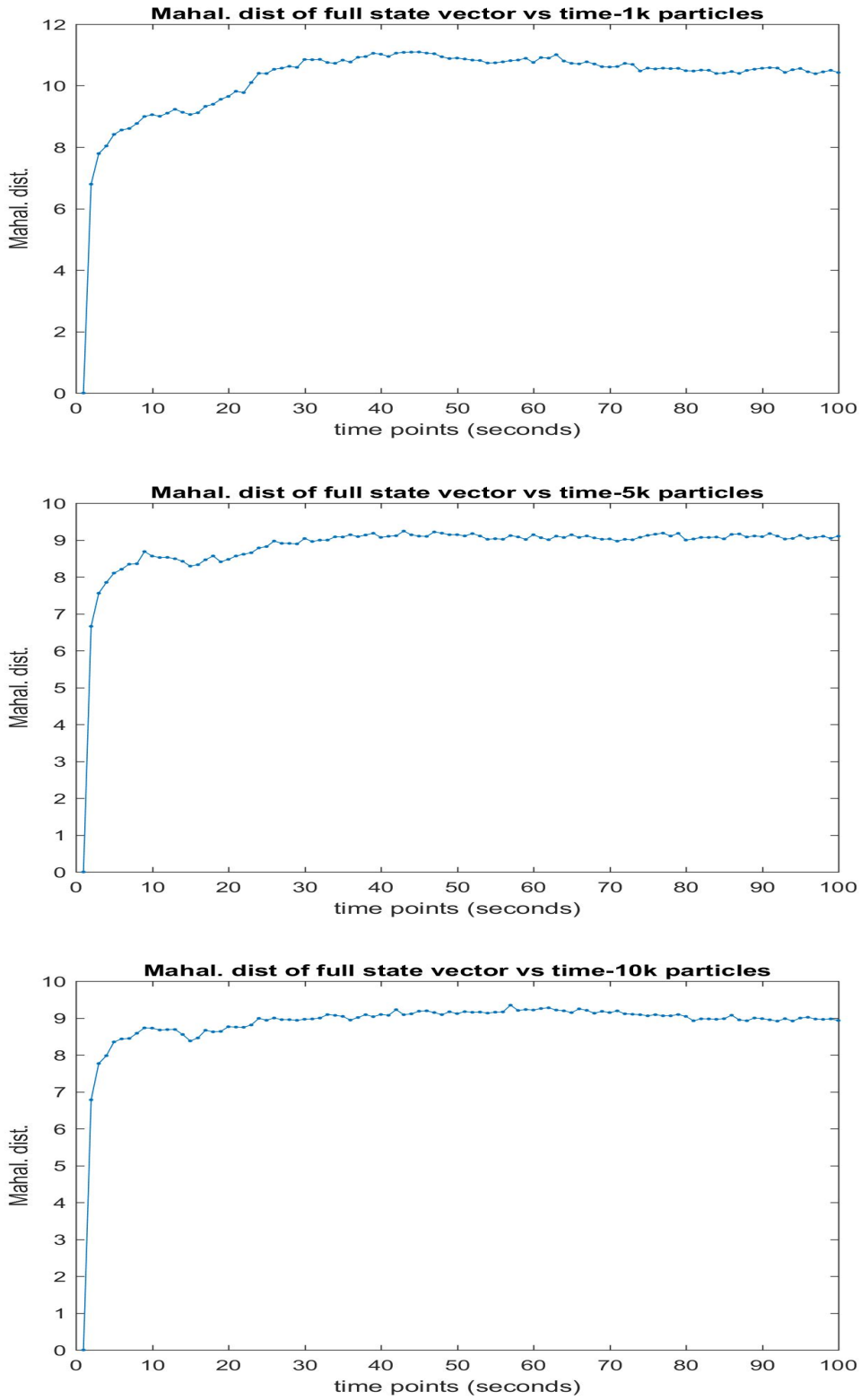

FIGURE 4.2: Plots of Mahalanobis distance over time points for various particle set sizes

As the plots in Figure 4.2 above show, the Mahalanobis distance does not converge to the dimension of the state space (i.e. 9) for particle set size = 1,000. However for particle set sizes = 5,000 and above, we face no such issues. Thus we choose 5,000 particles as our default particle set size. In all the following sections of this report, we assume that the number of particles are kept fixed at 5,000 unless otherwise stated.

In this chapter, we have given the details of the particle filter we have implemented that would be used to track the vehicles. In the next chapter, we look at the problem of tracking vehicles where all the vehicles pass through occlusion zones. We also look at the problem of inferring the true number of vehicles (based on their mutual *interactions*), when certain vehicles are totally occluded (i.e. hidden) from the sensor.

Chapter 5

Determining Number of Targets in the Presence of Occlusion

5.1 Introduction

In this chapter we add two layers of complexity to our model. In the first stage, we add occlusion zones in the path of the vehicles and see how well the particle filter performs in this situation. In the second stage, we add a further layer of complexity by assuming that in addition to vehicles passing through occlusion zones, we can have one (or more) vehicles being totally occluded (i.e. hidden). This means that the sensor fails to pick up measurements from a given vehicle, and in this sense the vehicle is occluded from the sensor. This might happen if say, a dark coloured car drives into a shadow of a building, then a vision based tracking algorithm similar to Aeschliman's system [5] based on the colour/shape/contour of the car fails to pick up the pixels related to that car when the surrounding is also dark, but in reality the vehicle is not obstructed from the sensor's field of view. In such cases the sensor may not register any measurements from that particular vehicle.

5.2 Experimental Setup - First Stage

The objective of this first stage is to add *occlusion zones* in the path of the vehicles and investigate how well we can track the targets using the particle filter. By an occlusion zone, we mean an interval $[a, b]$ such that when the vehicle positions are within the zone defined by the above interval, they are hidden from the sensor's field of view, and thus we do not get any measurements. However, once the vehicles emerge from the zone, the sensor again starts to pick up measurements. The zones defined as above represent distances, so when the *true* vehicle positions are within the interval defined above, they are hidden from the sensor.

In this stage, we assume two separate occlusion zones in two separate runs of the particle filter: **Occlusion Zone 1**: [100m, 150m] and **Occlusion Zone 2**: [300m, 400m]. Note that both the occlusion zones are *wide enough* such that there is a substantial time interval when *all* the vehicles are completely hidden from the sensor and we do not get *any* measurements at all. The total length travelled by the vehicles during the time interval of 100 seconds is approx. 500m and so all the vehicles have emerged from the second occlusion zone by the time the simulation ends. In the first experimental setup we have only Occlusion Zone 1, whereas in the second setup we have both Occlusion Zones 1 and 2.

5.2.1 Results

As before, we run the particle filter for 5,000 simulations (each simulation consisting of 100 seconds). The results are summarised below. For comparison

we also provide the results from the particle filter run without any occlusion zones:

Occlusion Zone	MSE: All vehicle positions	MSE: vehicle 1 position	MSE: vehicle 2 position	MSE: vehicle 3 position
NONE	2.2938	0.5104	0.6587	1.1247
[100m, 150m]	2.6587	0.5804	0.7536	1.3244
[100m, 150m] and [300m, 400m]	3.6244	0.8419	1.0221	1.7604

TABLE 5.1: MSE of errors in vehicle position estimates averaged over 5,000 simulations

As can be seen from the table above, if we compare the figures in rows 2 and 3 in the table (representing the two setups with occlusion zones) to those obtained for the particle filter run without any occlusion zone (row 1 in the table) we see that the MSE has increased substantially. Moreover, the wider the occlusion zones, the larger the MSE becomes, with the MSE increasing by approximately 15% for the setup with only one occlusion zone to 60% for the setup with two occlusion zones. The increase in MSE can also be seen in all the position estimates for each of the three vehicles. The results obtained are as expected as we have less information in the occluded cases since we do not receive any measurements when the vehicles pass through the occlusion zones. However, the MSE is still substantially lower than the measurement noise ($\sigma^2 = 9$) and so we see that the particle filter performs quite well in tracking the vehicles even in the presence of long periods of occlusion .

5.2.2 Conclusions

The results obtained in this stage have shown that the particle filter does a fairly good job of tracking the vehicles even when the vehicles pass through occlusion zones. Moreover, when the vehicles are occluded from the sensor, the

particle filter still allows us to make estimates of the state vectors (including vehicle positions) and so we do not lose all information of the vehicle states. This was one of the main drawbacks of using tracklets [17]. Secondly, throughout this stage we have assumed that we know the number of vehicles that we are tracking (three in this case). In the next stage, we relax this assumption, and use the particle filter to decide how many vehicles there are in reality.

5.3 Experimental Setup - Second Stage

In this stage we assume that we still have three vehicles (ground truth) and only one occlusion zone [100m, 200m]. In addition, we assume that vehicle 2 is *permanently* occluded (i.e. hidden from the sensor). The goal of this stage is to employ Bayesian model selection techniques to determine the true number of vehicles and then use the particle filter to track them as before. So even though the sensor recieves two measurements, we hope to deduce that in reality there are three vehicles. In this case, our assumption that vehicle 2 is occluded is a special case of any particular vehicle being occluded and in the next chapter we will further relax this assumption.

5.4 Theory of Bayesian Model Selection

The use of Bayes factors is a Bayesian alternative to classical hypothesis testing ([4], [1]) and can thus be used as a method for model selection. Given data $\mathcal{D}$, say we are comparing between a set of models $\left\{\mathcal{M}_{(i)}\right\}_{i=1}^{m}$. The natural way to compare between different models is via their probability

$$\mathbf{P}\left(\mathcal{M}_{(i)}|\mathcal{D}\right) = \frac{\mathbf{P}\left(\mathcal{D}|\mathcal{M}_{(i)}\right)\mathbf{P}\left(\mathcal{M}_{(i)}\right)}{\mathbf{P}\left(\mathcal{D}\right)}$$

To compare models, the denominator, which sums over the potentially huge space of all possible models, $\mathbf{P}(\mathcal{D}) = \sum_i \mathbf{P}\left(\mathcal{D}|\mathcal{M}_{(i)}\right)\mathbf{P}\left(\mathcal{M}_{(i)}\right)$ is not required. Prior preference for models can be included in $\mathbf{P}\left(\mathcal{M}_{(i)}\right)$. Assuming all models have equal prior preference, we have $\mathbf{P}\left(\mathcal{M}_{(i)}|\mathcal{D}\right) \propto \mathbf{P}\left(\mathcal{D}|\mathcal{M}_{(i)}\right)$

Given a model selection problem in which we have to choose between two models, on the basis of observed data $\mathcal{D}$, the plausibility of the two different models, say $\mathcal{M}_{(1)}$ and $\mathcal{M}_{(2)}$, parametrised by model parameter vectors, say θ_1 and θ_2 is assessed by the **Bayes factor** F given by:

$$\mathbf{F} = \frac{\mathbf{P}\left(\mathcal{D}|\mathcal{M}_{(1)}\right)}{\mathbf{P}\left(\mathcal{D}|\mathcal{M}_{(2)}\right)} = \frac{\int \mathbf{P}\left(\mathcal{D}|\theta_1, \mathcal{M}_{(1)}\right)\mathbf{P}\left(\theta_1|\mathcal{M}_{(1)}\right)d\theta_1}{\int \mathbf{P}\left(\mathcal{D}|\theta_2, \mathcal{M}_{(2)}\right)\mathbf{P}\left(\theta_2|\mathcal{M}_{(2)}\right)d\theta_2} \tag{5.1}$$

Unlike a likelihood-ratio test, this Bayesian model comparison does not depend on any single set of parameters, as it integrates over all parameters in each model (with respect to their respective priors). However, an advantage of the use of Bayes factors is that it automatically, and quite naturally, includes a penalty for including too much model structure or too many parameters. It thus guards against *overfitting* and this property has been called the *automatic Occam's Razor* ([18], [24]).

5.5 Computation of Bayes Factor

Say that we are comparing two models, $\mathcal{M}_{(2)}$ representing 2 vehicles vs. $\mathcal{M}_{(3)}$ representing 3 vehicles. We only observe the leading and trailing vehicles (i.e. vehicles 1 and 3) and hence only receive two measurements. There may/may not be another middle vehicle present. We want to employ model selection to deduce if another vehicle exists, i.e. given the observations, which out of the models $\mathcal{M}_{(2)}$ or $\mathcal{M}_{(3)}$ is more probable. Let $\mathbf{Z}_{1:\tau} = \{\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_\tau\}, 1 \leq \tau \leq T$

denote the observations (or data) upto and including time τ, where T is the total time period of available observations. Also let $\mathbf{X}_{1:\tau} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_\tau\}, 1 \leq \tau \leq T$ denote the particle filter estimates of the state space vectors upto and including time τ.

We have:

$$\begin{aligned}
\mathbf{P}\left(\mathbf{Z}_{1:\tau}|\mathcal{M}_{(i)}\right) &= \int \mathbf{P}\left(\mathbf{Z}_{1:\tau}, \mathbf{X}_{1:\tau}|\mathcal{M}_{(i)}\right) d\mathbf{X}_1 d\mathbf{X}_2 \ldots d\mathbf{X}_\tau \\
&= \int \mathbf{P}\left(\mathbf{Z}_{1:\tau}|\mathbf{X}_{1:\tau}, \mathcal{M}_{(i)}\right) \mathbf{P}\left(\mathbf{X}_{1:\tau}|\mathcal{M}_{(i)}\right) d\mathbf{X}_1 d\mathbf{X}_2 \ldots d\mathbf{X}_\tau \\
&= \int \prod_{t=1}^{\tau} \left[\mathbf{P}\left(\mathbf{Z}_t|\mathbf{X}_t, \mathcal{M}_{(i)}\right) \mathbf{P}\left(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathcal{M}_{(i)}\right) d\mathbf{X}_t\right] \\
&= \int \left[\prod_{t=1}^{\tau-1} \mathbf{P}\left(\mathbf{Z}_t|\mathbf{X}_t, \mathcal{M}_{(i)}\right) \mathbf{P}\left(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathcal{M}_{(i)}\right) d\mathbf{X}_t\right] \\
&\times \left[\int \mathbf{P}\left(\mathbf{Z}_\tau|\mathbf{X}_\tau, \mathcal{M}_{(i)}\right) \mathbf{P}\left(\mathbf{X}_\tau|\mathbf{X}_{\tau-1}, \mathcal{M}_{(i)}\right) d\mathbf{X}_\tau\right] \qquad (5.2)
\end{aligned}$$

It is difficult to compute the above expression analytically as we do not have a closed form expression for $\mathbf{P}\left(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathcal{M}_{(i)}\right)$ (See 3.3 and 3.4). Hence we resort to Monte Carlo sampling. Say we denote a function $f\left(\mathbf{X}_\tau|\mathcal{M}_{(i)}\right) = \mathbf{P}\left(\mathbf{Z}_\tau|\mathbf{X}_\tau, \mathcal{M}_{(i)}\right)$. Then the integral $\int \mathbf{P}\left(\mathbf{Z}_\tau|\mathbf{X}_\tau, \mathcal{M}_{(i)}\right) \mathbf{P}\left(\mathbf{X}_\tau|\mathbf{X}_{\tau-1}, \mathcal{M}_{(i)}\right) d\mathbf{X}_\tau$ is equivalent to finding $\mathbf{E}\left[f(\mathbf{X}_\tau|\mathcal{M}_{(i)})\right]$ where the expectation is with respect to the distribution

$\mathbf{P}\left(\mathbf{X}_\tau|\mathbf{X}_{\tau-1}, \mathcal{M}_{(i)}\right)$. So if we have samples $\mathbf{X}_\tau^{(1)}, \mathbf{X}_\tau^{(2)}, \ldots, \mathbf{X}_\tau^{(N)} \sim \mathbf{P}\left(\mathbf{X}_\tau|\mathbf{X}_{\tau-1}, \mathcal{M}_{(i)}\right)$, then we have:

$$\mathbf{E}\left[f(\mathbf{X}_\tau|\mathcal{M}_{(i)})\right] \approx \frac{1}{N} \sum_{n=1}^{N} f\left(\mathbf{X}_\tau^{(n)}|\mathcal{M}_{(i)}\right) \qquad (5.3)$$

However, note that since we are using a particle filter, hence given a model $\mathcal{M}_{(i)}$, we already have particles $\left\{\mathbf{X}_\tau^{(n)}\right\}_{n=1}^{N} \sim \mathbf{P}\left(\mathbf{X}_\tau|\mathbf{X}_{\tau-1}, \mathcal{M}_{(i)}\right)$ (See 4.2.2).

Thus in 5.2, using the above approximation, we can write:

$$\begin{aligned}
\mathbf{P}\left(\mathbf{Z}_{1:\tau}|\mathcal{M}_{(i)}\right) &= \int \left[\prod_{t=1}^{\tau-1} \mathbf{P}\left(\mathbf{Z}_t|\mathbf{X}_t, \mathcal{M}_{(i)}\right) \mathbf{P}\left(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathcal{M}_{(i)}\right) d\mathbf{X}_t\right] \\
&\qquad \times \mathbf{E}\left[f(\mathbf{X}_\tau|\mathcal{M}_{(i)})\right] \\
\implies \mathbf{P}\left(\mathbf{Z}_{1:\tau}|\mathcal{M}_{(i)}\right) &= \mathbf{P}\left(\mathbf{Z}_{1:\tau-1}|\mathcal{M}_{(i)}\right) \times \mathbf{E}\left[f(\mathbf{X}_\tau|\mathcal{M}_{(i)})\right] \\
\implies \mathbf{P}\left(\mathbf{Z}_{1:\tau}|\mathcal{M}_{(i)}\right) &\approx \mathbf{P}\left(\mathbf{Z}_{1:\tau-1}|\mathcal{M}_{(i)}\right) \times \left(\frac{1}{N}\sum_{n=1}^{N} f\left(\mathbf{X}_\tau^{(n)}|\mathcal{M}_{(i)}\right)\right) \qquad (5.4)
\end{aligned}$$

This gives us a recursive relation to calculate $\mathbf{P}\left(\mathbf{Z}_{1:\tau}|\mathcal{M}_{(i)}\right)$. Say we are comparing between the models $\mathcal{M}_{(2)}$ and $\mathcal{M}_{(3)}$. Using 5.1, we can now write the Bayes factor $\mathbf{F}_\tau$ at time τ as:

$$\begin{aligned}
\mathbf{F}_\tau &= \frac{\mathbf{P}\left(\mathbf{Z}_{1:\tau}|\mathcal{M}_{(2)}\right)}{\mathbf{P}\left(\mathbf{Z}_{1:\tau}|\mathcal{M}_{(3)}\right)} \\
\implies \mathbf{F}_\tau &\approx \frac{\mathbf{P}\left(\mathbf{Z}_{1:\tau-1}|\mathcal{M}_{(2)}\right)}{\mathbf{P}\left(\mathbf{Z}_{1:\tau-1}|\mathcal{M}_{(3)}\right)} \times \left[\frac{\sum_{n=1}^{N} f\left(\mathbf{X}_\tau^{(n)}|\mathcal{M}_{(2)}\right)}{\sum_{n=1}^{N} f\left(\mathbf{X}_\tau^{(n)}|\mathcal{M}_{(3)}\right)}\right] \\
\implies \mathbf{F}_\tau &\approx \mathbf{F}_{\tau-1} \times \left[\frac{\sum_{n=1}^{N} f\left(\mathbf{X}_\tau^{(n)}|\mathcal{M}_{(2)}\right)}{\sum_{n=1}^{N} f\left(\mathbf{X}_\tau^{(n)}|\mathcal{M}_{(3)}\right)}\right] \qquad (5.5)
\end{aligned}$$

Thus this gives us an easy recursive relation to calculate the Bayes Factor for comparison between two models.

Finally, note that the expression for $\frac{1}{N}\sum_{n=1}^{N} f\left(\mathbf{X}_\tau^{(n)}|\mathcal{M}_{(i)}\right)$ simplifies as follows:

$$\frac{1}{N}\sum_{n=1}^{N} f\left(\mathbf{X}_\tau^{(n)}|\mathcal{M}_{(i)}\right) = \frac{1}{N}\sum_{n=1}^{N} \mathbf{P}\left(\mathbf{Z}_\tau|\mathbf{X}_\tau^{(n)}, \mathcal{M}_{(i)}\right) \qquad (5.6)$$

But $\mathbf{P}\left(\mathbf{Z}_\tau|\mathbf{X}_\tau^{(n)}, \mathcal{M}_{(i)}\right) \forall n \in \{1, \ldots, N\}$ are precisely the importance weights calculated in the measurement update step of the particle filter. (See 4.2.2).

Hence, $\frac{1}{N}\sum_{n=1}^{N} f\left(\mathbf{X}_{\tau}^{(n)}|\mathcal{M}_{(i)}\right)$ is simply the sum of the importance weights calculated in the measurement update step of the particle filter, given a particular model.

The above derivation results in an expression very similar to that obtained in [35], where the authors do not specifically use a Bayes factor methodology but the goal is similar in the sense of using a particle filter to perform recursive model selection in a Bayesian framework.

5.5.1 Implementation Details

As stated earlier in 5.3, in this stage we assume that the ground truth is represented by 3 vehicles, but vehicle 2 is *permanently* occluded (i.e. hidden from the sensor). There is only one **Occlusion Zone** [100m, 200m] in this stage. We initialize two particle filters at time $t = 1$, one with state space $\in \mathbb{R}^6$ (representing 2 vehicles) and another with state space $\in \mathbb{R}^9$ (representing 3 vehicles). After that we let the particle filter run and compute the Bayes Factor recursively as above, at each time step. We expect that even though there are no measurements coming from vehicle 2, its effect on vehicle 3 would manifest itself via the the interaction between vehicles 2 and 3. Hence after sufficient number of observations have come in, the Bayes factor should indicate that model $\mathcal{M}_{(3)}$ with 3 vehicles is more plausible, given the data. We present the results below in the next section.

5.5.2 Results

We present below some sample cases which shows the Bayes factor based model selection at work.

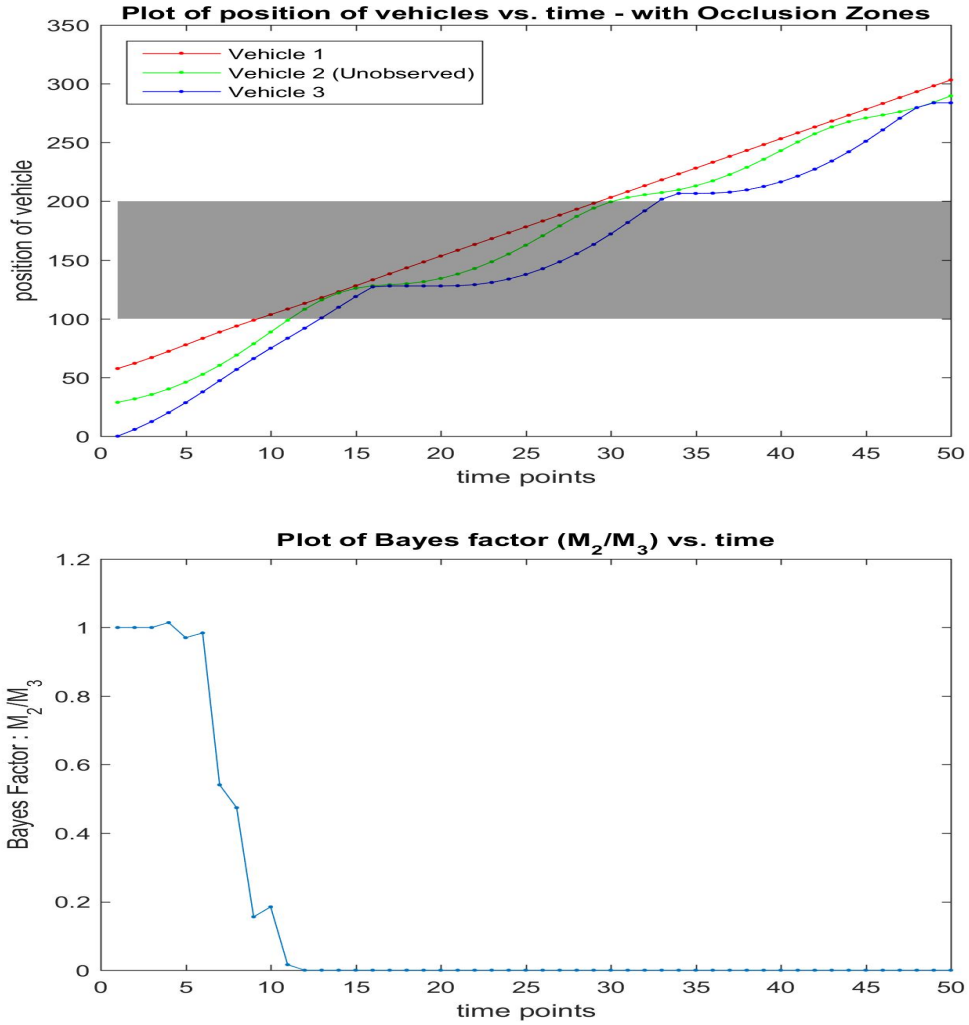

FIGURE 5.1: Bayes factor model selection : Example 1

In the top plot above in Figure 5.1, we see the trajectories of the 3 vehicles (ground truth). Note that vehicle 2 (green line) does not generate any measurements. The dark shaded area represent the the occlusion zone. The Bayes factor at $t = 1$ starts from 1, as before any measurements are received, both models are equally probable and so the ratio as given by 5.1 is 1. We see that the Bayes factor converges to 0 (i.e. $\mathcal{M}_{(3)}$ is much more probable than $\mathcal{M}_{(2)}$) before the vehicles enter the occlusion zone. If we notice how the Bayes factor

value changes over time, we see that it starts to steadily favour $\mathcal{M}_{(3)}$ from time point 7 onwards. The top plot shows that from time point 7 onwards, vehicle 3 approaches close to vehicle 2 (which is unobserved), and hence the braking effect kicks in. It is this *interaction* that is being captured by the Bayes factor. Under the model $\mathcal{M}_{(2)}$ in which we assume only vehicle 1 and vehicle 3 exist, there should be no reason for vehicle 3 to slow down as the separation distance between vehicle 1 and vehicle 3 is still large at this stage. Hence we see that Bayesian model selection is able to make successful inference of the existence of the middle vehicle (i.e. vehicle 2).

We present below some further examples of the working of the Bayes factor.

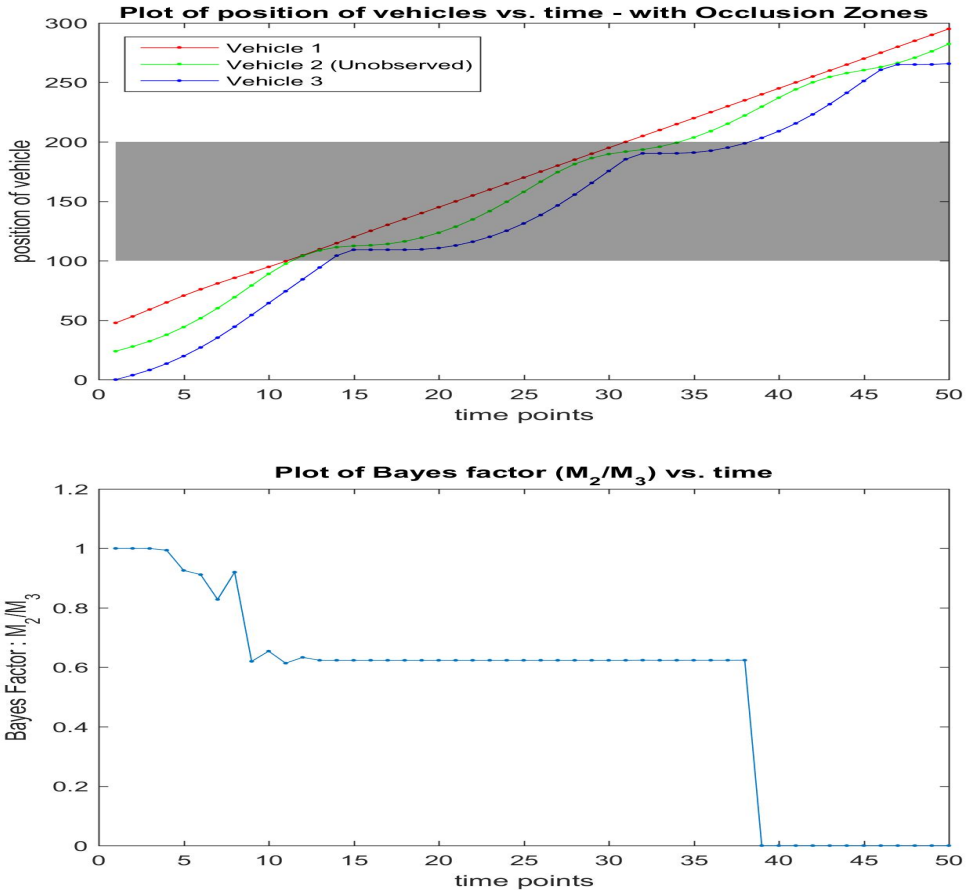

FIGURE 5.2: Bayes factor model selection : Example 2

In Figure 5.2 above we see that the Bayes factor does not converge conclusively in favour of either model before the vehicles enter the occlusion zone. While the vehicles are in the occlusion zone, (since no measurements are received) the Bayes factor stays constant. However, once the vehicles leave the occlusion zone, the Bayes factor quickly converges to 0. This example demonstrates that a sufficient number of *informative* observations (i.e. vehicle interactions) are required before Bayesian model selection can yield a definitive result. We finish with one final example:

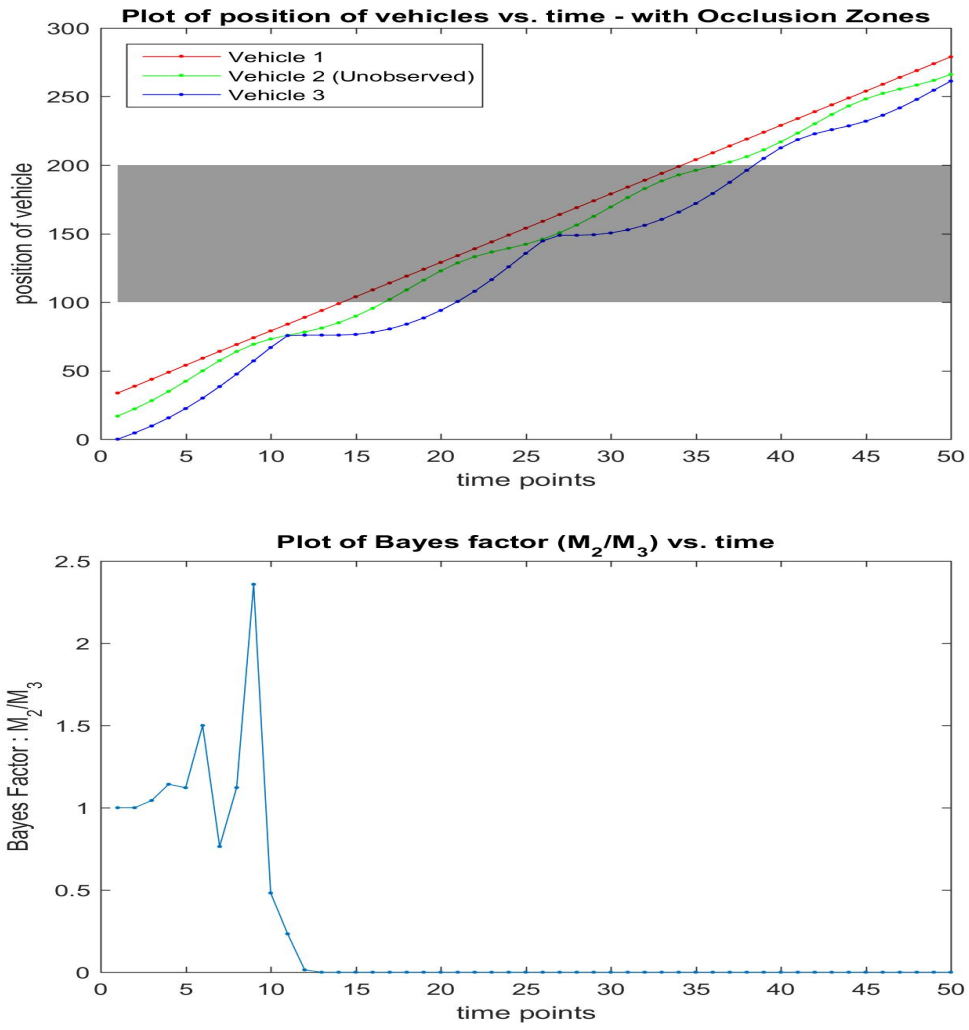

FIGURE 5.3: Bayes factor model selection : Example 3

The last example Figure 5.3 above shows that the Bayes factor can oscillate a lot before converging to a definitive value. Note that our computation of the Bayes factor is only an approximation (5.5), and hence it is possible that the approximation quality may not be appropriate for all scenarios. The hope is that if we have enough observations, the error in the approximation would not have any material effect on model selection. We actually ran the Bayes factor model section algorithm for 10,000 simulations and in *all* the simulations, the Bayes factor indicated that the model $\mathcal{M}_{(3)}$ representing 3 vehicles is more plausible than $\mathcal{M}_{(2)}$ representing 2 vehicles.

We also experimented with model section on a similar scenario as above but this time the ground truth being 2 vehicles and NO vehicle being permanently occluded (i.e. hidden from the sensor). We then ran model section via the Bayes factor between $\mathcal{M}_{(2)}$ representing 2 vehicles and $\mathcal{M}_{(3)}$ representing 3 vehicles as before. For the model $\mathcal{M}_3$ with 3 vehicles, we assumed, in turn, that there are in reality 3 vehicles, but vehicle 1 is permanently occluded. Similarly, on a next experiment, for $\mathcal{M}_{(3)}$ we assumed that vehicle 2 is permanently occluded. In this case also, for all the 10,000 simulations and the above cases, the Bayes factor favoured the model $\mathcal{M}_{(2)}$ = 2 vehicles. (To clarify, we did not consider the model $\mathcal{M}_{(3)}$ with vehicle 3 being permanently occluded, as in this case, there is no way for our model selection algorithm to decide if vehicle 3 exists, since it being permanently occluded, we cannot observe any effect of its existence by only observing measurements from vehicles 1 and 2).

5.5.3 Conclusions

In this chapter we have shown that using a particle filter in conjunction with Bayesian model section, one can make inferences about the true number of vehicles (as long as the vehicles are close enough to *interact*) even if one of them is permanently occluded or hidden from the sensor. This allows us to dynamically determine the true number of targets. The method presented above is general and can be readily extended to select among multiple models simultaneously, say choosing between 2, 3 and 4 vehicles when the sensor receives, say, two measurements at each time point. In the next chapter, we make an extension to this multiple model selection paradigm. We allow *new* vehicles to come in or *existing* vehicles to be removed from our target group while they pass through the occlusion zone. Our aim is to perform dynamic model selection to determine the changed number of vehicles and be able to re-capture the vehicles in the *original* target group after they emerge from the occlusion zone as long as they exist and even if new *previously unseen* vehicles are interspersed within the original target vehicles.

Finally, note that we have assumed that there are no mis-detections of the vehicles. This phenomena is called *clutter*. Exploring how the above methods work in the presence of clutter can form a part of future extension of this work.

Chapter 6

Multi-Target Tracking with Changing Number of Targets

6.1 Introduction

In this final stage we extend our problem to additionally incorporate *events* in which vehicles appear/disappear *inside* the occlusion zones and thus the occurrence of the event is invisible to the sensor. Before the vehicles disappear into the occlusion zones, we will use a particle filter in conjunction with Bayesian model selection to determine the true number of vehicles as well as simultaneously track them.

Once the vehicles come out of the occlusion zone, *post* an event of vehicle appearance/disappearance, we will again use Bayesian model selection to dynamically determine the number of vehicles emerging from the occlusion zone. In this way we infer if one (or more) of the vehicles being tracked have been lost or if new vehicles have appeared. Additionally, we also want to make predictions of which vehicle has disappeared or is new in the group we are tracking. As a final task, we aim to make predictions as to *where* in the occlusion zone the event of appearance/disappearance possibly took place.

Thus the goal is not only to provide predictions on the state of the vehicles inside the occlusion zones, but also to make inferences on which vehicles have come in or gone out of our group and where the event took place. This also means that our tracker should be able to re-capture the vehicles in the *original* target group after they emerge from the occlusion zone as long as they exist and even if new *previously unseen* vehicles are interspersed within the original target vehicles.

6.2 Experimental Setup

For this stage we run a set of three experiments. In all of them there is only one occlusion zone of [100m, 150m]. Each simulation run consists of 50 time points (i.e. 50 seconds), so the vehicles travel approximately 250m in each simulation. Also by an *event* we mean either one vehicle disappears, i.e. does not come out of the occlusion zone or a new vehicle gets added to the group of vehicles we are tracking. Such an event always occurs inside the occlusion zone, so that it is not observed by the sensor. Also note that the width of the occlusion zone means that for approximately 9-10 seconds, *none* of the vehicles are visible. The experiments are described below:

1. **Setup 1**: *Ground truth*: We start with 3 vehicles and they pass through the occlusion zone and *all* three come out unaffected from the zone. No vehicle gets added or deleted, i.e. *no event* takes place in the occlusion zone.

2. **Setup 2**: *Ground truth*: We start with 3 vehicles and while passing through the occlusion zone, one vehicle *disappears* and thus only two vehicles come out. We consider 3 sub cases of this setup as below:

(a) **Setup2a**: *Ground truth*: In the occlusion zone, vehicle 1 disappears, so vehicles 2 and 3 come out of the zone.

(b) **Setup2b**: *Ground truth*: In the occlusion zone, vehicle 2 disappears, so vehicles 1 and 3 come out of the zone.

(c) **Setup2c**: *Ground truth*: In the occlusion zone, vehicle 3 disappears, so vehicles 1 and 2 come out of the zone.

3. **Setup 3**: *Ground truth*: We start with 3 vehicles and while passing through the occlusion zone one vehicle *gets added* to the group, and so four vehicles come out. As before, we consider 4 sub cases of this setup as below:

(a) **Setup3a**: *Ground truth*: In the occlusion zone, a vehicle gets added in front of vehicle 1, so when the four vehicles come out, the first vehicle in the group is the *new* vehicle.

(b) **Setup3b**: *Ground truth*: In the occlusion zone, a vehicle gets added between vehicles 1 and 2, so when the four vehicles come out, the second vehicle in the group is the *new* vehicle.

(c) **Setup3c**: *Ground truth*: In the occlusion zone, a vehicle gets added between vehicles 2 and 3, so when the four vehicles come out, the third vehicle in the group is the *new* vehicle.

(d) **Setup3d**: *Ground truth*: In the occlusion zone, a vehicle gets added behind vehicle 3, so when the four vehicles come out, the last vehicle in the group is the *new* vehicle.

In all the cases above, where the ground truth is to simulate an event (vehicle appearance or disappearance), we randomly select a point within the occlusion zone in which to simulate the event. For example, if the ground truth is

to add a vehicle between vehicles 2 and 3, then we randomly choose a distance d (within the occlusion zone) and then as soon as vehicle 3 crosses the distance d, we simulate initialisation of another vehicle between the current vehicle 2 and vehicle 3 positions, ensuring that the new vehicle's *starting x-position* is *also* within the occlusion zone. This starting x-position is defined as the *distance of the event* in this case. To initialise a new vehicle in the occlusion zone, we randomly choose their x-position, velocity and acceleration, such that they satisfy all the constraints laid out in Chapter 3. Hence, from this time point onwards, the simulator models the dynamics of the four vehicles under the usual assumptions of *interactions* between two consecutive vehicles.

Similarly, where the ground truth is for a vehicle (say vehicle 1) to disappear, then as before, we randomly choose a distance d (within the occlusion zone) and then as soon as vehicle 1 crosses the distance d, we delete its existence by removing all of its entries from the joint state vector of all the three vehicles. The x-position of the vehicle 1 which is removed at this time point is defined as the distance of the event in this case. Thus, from this time point onwards, vehicle 2 becomes the lead vehicle and thus operates under no constraints (i.e. independent motion). The original vehicle 3, which now becomes vehicle 2, is still constrained by the vehicle before it.

A simplification we have assumed above is that an event can consist only of addition or removal of *one* vehicle and no more, i.e. multiple vehicles cannot be removed or added in an event. Similarly, we also assume that multiple events (i.e. addition followed by removal of vehicles) are not permitted in a single occlusion zone. Both of the above are simplifications made in order to keep the computational costs low. The method underlying our analysis can

readily be extended to incorporate multiple events or one event affecting multiple vehicles, but at the expense of increased computational costs.

Finally note that, in all the cases above, since our tracker does not know if an event will or will not happen in the occlusion zone, hence when the vehicles emerge from the occlusion zone, it must make a model selection to determine the true number of vehicles. If the tracker determines that the number of vehicles has changed from what it was before entering the occlusion zone, it must then make a prediction as to which vehicle has disappeared or is new. As a added benefit of this prediction, it would also generate a prediction on *where* (i.e. at what distance) the event took place.

6.3 Methodology

In this section, we describe the underlying methodology employed. As before, we make some simplifying assumptions to reduce the computational complexity. We assume throughout this project, that there are no false vehicle detections. Hence, if the sensor receives N_1 measurements, we know that the number of vehicles has to be at least N_1 . For the purposes of this project, we assume that at most one vehicle can be permanently occluded (i.e. hidden from the sensor), and so if we receive N_1 measurements, we perform a model selection between N_1 and $N_1 + 1$ vehicles. This can easily be extended to perform a model selection between $N_1, N_1 + 1, N_1 + 2, \ldots$ etc. vehicles, (i.e. assuming more than one vehicle can be permanently occluded) at the expense of increased computational costs.

6.3.1 Underlying Theory

Stage 1 ***Before entering the occlusion zone*** **:** In this stage we have 3 vehicles (ground truth) and the problem is exactly the same as in 5.5.1. We define some terminology below:

Since we assume that there are no false detections, at any point of time in this stage, we get a maximum of $N_1 = 3$ measurements. So we perform a model selection between $\mathcal{M}_{(N_1)}^{(stage1)}$ representing N_1 vehicles vs. $\mathcal{M}_{(N_1+1)}^{(stage1)}$ representing $N_1 + 1$ vehicles in a similar fashion as described in 5.5.1.

Once we have determined how many vehicles there are (via the Bayes factor), we drop the other competing model, and assume that only the model that has been selected as optimal, is the truth. Call the selected model as $\mathcal{M}_{opt}^{(stage1)}$ which denotes that the true number of vehicles in this stage is say, $N_{opt}^{(stage1)}$. The particle filter associated with this selected model then is run till the time point just before any of the vehicles enter the occlusion zone.

Stage 2 ***After emerging from the occlusion zone*** **:** The first goal in this stage is to determine how many vehicles emerge from the occlusion zone. To achieve this, we again look at the number of measurements we obtain after all the vehicles have emerged from the occlusion zone. (Since we know the width of the occlusion zone and the average target speed of the vehicles (5 m/s), we can *wait* for a sufficiently long time after the vehicles disappear into the occlusion zone, such that we are certain that post this *waiting time*, all vehicles would have emerged from the occlusion zone).

Say the number of measurements we receive is N_2. We now need to perform a model selection between $\mathcal{M}_{(N_2)}^{(stage2)}$ representing N_2 vehicles vs. $\mathcal{M}_{(N_2+1)}^{(stage2)}$ representing $N_2 + 1$ vehicles. However, we do not have starting values to initialise the particle filters for the above models. Thus we proceed as follows.

For the discussion below, let $\mathcal{M}_{(N_2)}^{(stage2)}$ be the model under consideration.

1. **Case 1**: $N_2 = N_{opt}^{(stage1)} - 1$, i.e. the model under consideration represents the hypothesis that there is one vehicle *less* in the group that entered the occlusion zone. In this case we need to consider one model for each case, in turn, that each one of the $N_{opt}^{(stage1)}$ vehicles have been removed. For example, if $N_{opt}^{(stage1)} = 3$ and $N_2 = 2$, we get three cases similar to setups 2a, 2b and 2c in 6.2. Continuing with the above example, we define the following:

   $\mathcal{M}_{(N_2,i)}^{(stage2)} \rightarrow$ model denoting vehicle i is removed in occlusion zone in stage 2

   For example:

   $\mathcal{M}_{(N_2,1)}^{(stage2)} \rightarrow$ model denoting vehicle 1 is removed in occlusion zone in stage 2

   $T_1 \rightarrow$ last time point before *any* of the vehicles enter the occlusion zone

   $T_2 \rightarrow$ last time point before *all* the vehicles enter the occlusion zone and we stop receiving measurements

   $T_3 \rightarrow$ first time point when *any* of the vehicles emerge from the occlusion zone and measurements resume

   $T_4 \rightarrow$ first time point when *all* of the vehicles emerge from the occlusion zone

   Note that since we know (in this particular case) $N_2 = N_{opt}^{(stage1)} - 1$, this means the $\mathbf{X}$ state vector has undergone a change in dimension (from $\mathbb{R}^{3\times N_{opt}^{(stage1)}}$ to $\mathbb{R}^{3\times N_2}$) sometime between time points T_1 and T_4 due to an event. Let the time point at which the event occurs be denoted by T^*, where $T^* \in [T_1 + 1, T_4 - 1]$. For a given value of T^*, the model $\mathcal{M}_{(N_2,i)}^{(stage2)}$ for that value of T^*, is denoted by $\mathcal{M}_{(N_2,i,T^*)}^{(stage2)}$. Thus we have:

$\mathcal{M}^{(stage2)}_{(N_2,i,T^*)} \rightarrow$ model denoting vehicle i is removed in occlusion zone at time T^*
Note that we only get measurements from time points $1\colon T_2$ and from $T_3\colon \tau$.

Now for a given model $\mathcal{M}^{(stage2)}_{(N_2 i,T^*)}$ we need the following expression to calculate the Bayes factor (Note: In the expressions below model $\mathcal{M}^{(stage1)}_{(opt)}$ is already determined).

$$
\begin{aligned}
&\mathbf{P}\left(\mathbf{Z}_{1:T_2}, \mathbf{Z}_{T_3:\tau} | \mathcal{M}^{(stage1)}_{(opt)}, \mathcal{M}^{(stage2)}_{(N_2,i,T^*)}\right) \\
&= \int \mathbf{P}\left(\mathbf{Z}_{1:T_2}, \mathbf{Z}_{T_3:\tau}, \mathbf{X}_{1:T_2}, \mathbf{X}_{T_3:\tau} | \mathcal{M}^{(stage1)}_{(opt)}, \mathcal{M}^{(stage2)}_{(N_2,i,T^*)}\right) d\mathbf{X}_1 \ldots d\mathbf{X}_{T_2} d\mathbf{X}_{T_3} \ldots d\mathbf{X}_\tau \\
&= \int \mathbf{P}\left(\mathbf{Z}_{1:T_2}, \mathbf{Z}_{T_3:\tau} | \mathbf{X}_{1:T_2}, \mathbf{X}_{T_3:\tau}, \mathcal{M}^{(stage1)}_{(opt)}, \mathcal{M}^{(stage2)}_{(N_2,i,T^*)}\right) \\
&\quad \times \mathbf{P}\left(\mathbf{X}_{1:T_2}, \mathbf{X}_{T_3:\tau} | \mathcal{M}^{(stage1)}_{(opt)}, \mathcal{M}^{(stage2)}_{(N_2,i,T^*)}\right) d\mathbf{X}_1 \ldots d\mathbf{X}_{T_2} d\mathbf{X}_{T_3} \ldots d\mathbf{X}_\tau \\
&= \int \Bigg[\mathbf{P}\left(\mathbf{Z}_{1:T_1} | \mathbf{X}_{1:T_1}, \mathcal{M}^{(stage1)}_{(opt)}\right) \mathbf{P}\left(\mathbf{Z}_{T_1+1:T_2} | \mathbf{X}_{T_1+1:T_2}, \mathcal{M}^{(stage1)}_{(opt)}, \mathcal{M}^{(stage2)}_{(N_2,i,T^*)}\right) \\
&\quad \times \mathbf{P}\left(\mathbf{Z}_{T_3:\tau} | \mathbf{X}_{T_3:\tau}, \mathcal{M}^{(stage1)}_{(opt)}, \mathcal{M}^{(stage2)}_{(N_2,i,T^*)}\right) \mathbf{P}\left(\mathbf{X}_{1:T_1} | \mathcal{M}^{(stage1)}_{(opt)}\right) \\
&\quad \times \mathbf{P}\left(\mathbf{X}_{T_1+1:T_2}, \mathbf{X}_{T_3:\tau} | \mathbf{X}_{1:T_1}, \mathcal{M}^{(stage1)}_{(opt)}, \mathcal{M}^{(stage2)}_{(N_2,i,T^*)}\right) \Bigg] d\mathbf{X}_1 \ldots d\mathbf{X}_{T_2} d\mathbf{X}_{T_3} \ldots d\mathbf{X}_\tau \\
&= \int \Bigg[\left\{ \mathbf{P}\left(\mathbf{Z}_{1:T_1} | \mathbf{X}_{1:T_1}, \mathcal{M}^{(stage1)}_{(opt)}\right) \mathbf{P}\left(\mathbf{X}_{1:T_1} | \mathcal{M}^{(stage1)}_{(opt)}\right) \right\} \\
&\quad \times \mathbf{P}\left(\mathbf{Z}_{T_1+1:T_2} | \mathbf{X}_{T_1+1:T_2}, \mathcal{M}^{(stage1)}_{(opt)}, \mathcal{M}^{(stage2)}_{(N_2,i,T^*)}\right) \mathbf{P}\left(\mathbf{Z}_{T_3:\tau} | \mathbf{X}_{T_3:\tau}, \mathcal{M}^{(stage1)}_{(opt)}, \mathcal{M}^{(stage2)}_{(N_2,i,T^*)}\right) \\
&\quad \times \mathbf{P}\left(\mathbf{X}_{T_1+1:T_2}, \mathbf{X}_{T_3:\tau} | \mathbf{X}_{1:T_1}, \mathcal{M}^{(stage1)}_{(opt)}, \mathcal{M}^{(stage2)}_{(N_2,i,T^*)}\right) \Bigg] d\mathbf{X}_1 \ldots d\mathbf{X}_{T_2} d\mathbf{X}_{T_3} \ldots d\mathbf{X}_\tau
\end{aligned}
\tag{6.1}
$$

Note that the first term within the curly brackets in 6.1 above does not depend on $\mathcal{M}^{(stage2)}_{(N_2,i,T^*)}$, the model for which we are trying to calculate the Bayes factor for. Each of the other terms in 6.1 can be broken down into

terms before and after the event at T^*. We show below the calculations required to simplify 6.1 above:

$$\mathbf{P}\left(\mathbf{Z}_{1:T_1}\middle|\mathbf{X}_{1:T_1},\mathcal{M}_{(opt)}^{(stage1)}\right)\mathbf{P}\left(\mathbf{X}_{1:T_1}\middle|\mathcal{M}_{(opt)}^{(stage1)}\right)=f\left(\mathbf{X}_{1:T_1}\right)\text{ (say)}$$
$$\left(\text{term independent of }\mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right) \tag{6.2}$$

$$\begin{aligned}&\mathbf{P}\left(\mathbf{Z}_{T_1+1:T_2}\middle|\mathbf{X}_{T_1+1:T_2},\mathcal{M}_{(opt)}^{(stage1)},\mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right)\\&=\mathbf{P}\left(\mathbf{Z}_{T_1+1:\min(T_2,T^*-1)}\middle|T^*,\underbrace{\mathbf{X}_{T_1+1:\min(T_2,T^*-1)}}_{\mathbf{X}\text{ before event}},\mathcal{M}_{(opt)}^{(stage1)}\right)\\&\times\mathbf{P}\left(\mathbf{Z}_{\min(T_2,T^*-1)+1:T_2}\middle|\underbrace{\mathbf{X}_{\min(T_2,T^*-1)+1:T_2}}_{\mathbf{X}\text{ after event}},\mathcal{M}_{(opt)}^{(stage1)},\mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right)\end{aligned} \tag{6.3}$$

$$\begin{aligned}&\mathbf{P}\left(\mathbf{Z}_{T_3:\tau}\middle|\mathbf{X}_{T_3:\tau},\mathcal{M}_{(opt)}^{(stage1)},\mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right)\\&=\mathbf{P}\left(\mathbf{Z}_{T_3:T^*-1}\middle|T^*,\ \underbrace{\mathbf{X}_{T_3:T^*-1}}_{\mathbf{X}\text{ before event}},\mathcal{M}_{(opt)}^{(stage1)}\right)\\&\times\mathbf{P}\left(\mathbf{Z}_{\max(T_3,T^*):\tau}\middle|\underbrace{\mathbf{X}_{\max(T_3,T^*):\tau}}_{\mathbf{X}\text{ after event}},\mathcal{M}_{(opt)}^{(stage1)},\mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right)\end{aligned} \tag{6.4}$$

$$
\begin{aligned}
&\mathbf{P}\left(\mathbf{X}_{T_1+1:T_2}, \mathbf{X}_{T_3:\tau} \middle| \mathbf{X}_{1:T_1}, \mathcal{M}_{(opt)}^{(stage1)}, \mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right) \\
&\quad = \mathbf{P}\left(\underbrace{\mathbf{X}_{T_1+1:\min(T_2,T^*-1)}}_{\mathbf{X} \text{ before event}} \middle| T^*, \mathbf{X}_{1:T_1}, \mathcal{M}_{(opt)}^{(stage1)}\right) \\
&\qquad \times \mathbf{P}\left(\underbrace{\mathbf{X}_{\min(T_2,T^*-1)+1:T_2}}_{\mathbf{X} \text{ after event}} \middle| \mathbf{X}_{1:\min(T_2,T^*-1)}, \mathcal{M}_{(opt)}^{(stage1)}, \mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right) \\
&\qquad \times \mathbf{P}\left(\underbrace{\mathbf{X}_{T_3:T^*-1}}_{\mathbf{X} \text{ before event}} \middle| T^*, \mathbf{X}_{1:T_2}, \mathcal{M}_{(opt)}^{(stage1)}\right) \\
&\qquad \times \mathbf{P}\left(\underbrace{\mathbf{X}_{\max(T_3,T^*):\tau}}_{\mathbf{X} \text{ after event}} \middle| \mathbf{X}_{1:T_2}, \mathbf{X}_{\max(T_3,T^*-1):\tau}, \mathcal{M}_{(opt)}^{(stage1)}, \mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right)
\end{aligned} \tag{6.5}
$$

In all the expressions above, terms involving an empty interval, say for example $\mathbf{P}\left(\mathbf{Z}_{4:2}|...\right)$ are assumed to be unity. Plugging in the terms from 6.2, 6.3, 6.4 and 6.5 into 6.1 we get an expression involving $\mathbf{X}$ (before and after the event) and T^*. Our first task is to obtain the value of T^* for which $\mathbf{P}\left(\mathbf{Z}_{1:T_2}, \mathbf{Z}_{T_3:\tau}|\mathcal{M}_{(opt)}^{(stage1)}, \mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right)$ is maximum for a given value of i. Since $T^* \in [T_1+1, T_4-1]$, we try out each value of T^* in this range to get the optimum value of T^* such that:

$$
T_{N_2,i}^{*(opt)} = \underset{T^*}{\operatorname{argmax}}\, \mathbf{P}\left(\mathbf{Z}_{1:T_2}, \mathbf{Z}_{T_3:\tau}|\mathcal{M}_{(opt)}^{(stage1)}, \mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right) \tag{6.6}
$$

Hence we define:

$\mathcal{M}_{(N_2,i)}^{(stage2)} = \max_{T^*} \mathcal{M}_{(N_2,i,T^*)}^{(stage2)} \rightarrow$ model with highest likelihood among all models representing vehicle i being removed in the occlusion zone

For example:

$\mathcal{M}_{(N_2,1)}^{(stage2)} = \max_{T^*} \mathcal{M}_{(N_2,1,T^*)}^{(stage2)} \rightarrow$ model with highest likelihood among all models representing vehicle 1 being removed

Let:

$$i_{N_2}^{*(stage2)} = \underset{i}{\operatorname{argmax}}\, \mathcal{M}_{(N_2,i)}^{(stage2)} \tag{6.7}$$

Hence we define:

$$\mathcal{M}_{(N_2)}^{(stage2)} = \max_i \mathcal{M}_{(N_2,i)}^{(stage2)} = \max_i \max_{T^*} \mathcal{M}_{(N_2,i,T^*)}^{(stage2)}$$

Thus $\mathcal{M}_{(N_2)}^{(stage2)}$ represents the model which has the highest likelihood among all models that denote that at stage 2, N_2 is the true number of vehicles.

2. **Case 2**: $N_2 = N_{opt}^{(stage1)} + 1$, i.e. the model under consideration $M_{(N_2)}^{(stage2)}$ says that there is one vehicle *added* to the group that entered the occlusion zone. In this case, we proceed exactly as the case above, to find:

$$\mathcal{M}_{(N_2)}^{(stage2)} = \max_i \mathcal{M}_{(N_2,i)}^{(stage2)} = \max_i \max_{T^*} \mathcal{M}_{(N_2,i,T^*)}^{(stage2)}$$

where:

$\mathcal{M}_{(N_2,i)}^{(stage2)} \rightarrow$ model with highest likelihood among all models representing vehicle in ith position is *new* among the vehicles that emerge from the occlusion zone

$$i_{N_2}^{*(stage2)} = \underset{i}{\operatorname{argmax}}\, \mathcal{M}_{(N_2,i)}^{(stage2)} \tag{6.8}$$

In this case however, the dimension of $\mathbf{X}$ *increases* by 3 from what it was before the vehicles entered the occlusion zone.

3. **Case 3**: $N_2 = N_{opt}^{(stage1)}$, i.e. there is no change in vehicle numbers post passing through the occlusion zone. So to calculate the expression $\mathbf{P}\left(\mathbf{Z}_{1:T_2}, \mathbf{Z}_{T_3:\tau} | \mathcal{M}_{(opt)}^{(stage1)}, \mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right)$, we note that since $N_2 = N_{opt}^{(stage1)}$, we have $\mathcal{M}_{(N_2,i,T^*)}^{(stage2)} \equiv \mathcal{M}_{(N_2)}^{(stage2)}$, so there is no dependence on i or T^*. Thus in

this case, we simply run the particle filter associated with $\mathcal{M}_{(opt)}^{(stage1)}$ without any changes up until the end of the time period under consideration (i.e. τ). This would enable us to calculate $\mathbf{P}\left(\mathbf{Z}_{1:T_2}, \mathbf{Z}_{T_3:\tau} | \mathcal{M}_{(opt)}^{(stage1)}, \mathcal{M}_{(N_2,i,T^*)}^{(stage2)}\right)$.

The final step is to compare the models $\mathcal{M}_{(N_2)}^{(stage2)}$ and $\mathcal{M}_{(N_2+1)}^{(stage2)}$ via the Bayes factor methodology outlined in 5.5.1 to decide the true number of vehicles leaving the occlusion zone to give us $\mathcal{M}_{(opt)}^{(stage2)}$. Say $\mathcal{M}_{(opt)}^{(stage2)} \equiv \mathcal{M}_{(N_2)}^{(stage2)}$ is the model chosen above. Then, for this chosen N_2, we also have $i_{N_2}^{*(stage2)}$ from 6.7 or 6.8 which gives us which vehicle was removed if $N_2 = N_{opt}^{(stage1)} - 1$ or which vehicle is new if $N_2 = N_{opt}^{(stage1)} + 1$ respectively or to say *no* event took place if $N_2 = N_{opt}^{(stage1)}$.

Finally, note that if we select a model above for which $N_2 \neq N_{opt}^{(stage1)}$, then from 6.6 we have the optimal value of time point $T_{N_2,i}^{*(opt)}$ for given $i = i_{N_2}^{*(stage2)}$ which gives us at what time point the change in dimension of $\mathbf{X}$ took place. We thus run our particle filter from stage 1 corresponding to the chosen model ($\mathcal{M}_{(opt)}^{(stage1)}$) up until this time point. If the event is a vehicle removal, we simply read off the x-position of the vehicle affected at this time point from the state vector $\mathbf{X}$ and set this as our predicted distance of the event. If on the other hand, the event is a vehicle addition, we set the distance of the event as the starting x-position of the new vehicle corresponding to the model $\mathcal{M}_{(N_2,i,T^*)}^{(stage2)}$ where i and T^* are set to their optimal values as given above. This then gives us an estimate of the distance at which the event occurred inside the occlusion zone.

6.4 Implementation Details

We give below some details on how the values of the expressions in 6.2, 6.3, 6.4 and 6.5 are computed. First note that the expression in 6.2 is independent of $\mathcal{M}_{(N_2)}^{(stage2)}$ and hence would cancel out in the computation of Bayes factor between say models $\mathcal{M}_{(N_2)}^{(stage2)}$ and $\mathcal{M}_{(N_2+1)}^{(stage2)}$.

The expressions in 6.3 and 6.4 depend on choosing a particular value of $T^* \in [T_1 + 1, T_4 - 1]$ (time point when the dimension of $\mathbf{X}$ changes). Combining each individual term in 6.3 and 6.4 with the appropriate terms in 6.5, we see that each of them can be thought of as conditional expectations of the measurement likelihoods. Thus we can approximate them using similar methods used to approximate 5.2 by 5.3.

The only remaining detail is to clarify how we determine the state vector $\mathbf{X}$ for the appropriate particle filter at stage 2. In cases 1 and 2 above, once we fix a given value of T^* and i in $\mathcal{M}_{(N_2,i,T^*)}^{(stage2)}$, we let the particle filter corresponding to the model from stage 1 ($\mathcal{M}_{(opt)}^{(stage1)}$), run till time point T^*, and then modify appropriately the estimated state vector *at this time point*. By modifying the state vector, we mean that if the dimension of $\mathbf{X}$ *decreases*, we then delete the appropriate entries of $\mathbf{X}$ corresponding to the vehicle which we assume is removed, or if the dimension of $\mathbf{X}$ *increases*, then we randomly initialise the x-position, velocity and acceleration of a new vehicle at that time point (satisfying the constraints in Chapter 3) and add these to the state vector $\mathbf{X}$ at the appropriate location corresponding to where we assume the new vehicle is in relation to the original group of vehicles we were tracking.

Finally, we clarify how we select which models to consider when employing the Bayes factor model selection methodology. For stage 1, where we receive $N_1 = 3$ measurements (say), we do a model selection between 3 vehicles

($\mathcal{M}_{(3)}^{(stage1)}$) vs. 4 vehicles ($\mathcal{M}_{(4)}^{(stage1)}$). For the model representing 4 vehicles, we select the *best* model (via Bayes factor) among all the models of 4 vehicles, where the models represent, in turn, cases where one particular vehicle is assumed to be totally occluded (i.e. hidden from the sensor) and the remaining three vehicles correspond to the three measurements received. The model selection is then performed between the model of 3 vehicles ($\mathcal{M}_{(3)}^{(stage1)}$) vs. the *best* model among all the models representing 4 vehicles to give us $\mathcal{M}_{(opt)}^{(stage1)}$.

For stage 2, we receive N_2 measurements leading us to compare between $\mathcal{M}_{(N_2)}^{(stage2)}$ vs. $\mathcal{M}_{(N_2+1)}^{(stage2)}$. Say $N_2 = 2$, so we do a model section between 2 vehicles ($\mathcal{M}_{(2)}^{(stage2)}$) vs. 3 vehicles ($\mathcal{M}_{(3)}^{(stage2)}$). If $\mathcal{M}_{(opt)}^{(stage1)}$ from stage 1 represents a model of 3 vehicles, then there is only one model in $\mathcal{M}_{(3)}^{(stage2)}$, as this assumes that no event has occurred in the occlusion zone. However for this case, $\mathcal{M}_{(2)}^{(stage2)}$ represents the *best* model (according to Bayes factor) among all the models of 2 vehicles, where the models represent, in turn, cases where one particular vehicle is assumed to be have *disappeared* in the occlusion zone and the remaining two vehicles correspond to the two measurements received. Thus the model selection is then performed between the model of 3 vehicles ($\mathcal{M}_{(3)}^{(stage2)}$) vs. the *best* model among all the models representing 2 vehicles to give us $\mathcal{M}_{(opt)}^{(stage2)}$. For other cases, for example, when the event leads to a vehicle *appearance* in stage 2, the range of models to consider are determined in a similar fashion.

6.5 Results

We ran 1,000 simulations for each scenario above (setups 1 - 3d) and collected the instances where our method failed to detect the correct number of vehicles or once the correct number of vehicles have been detected, the method failed to correctly identify the true vehicle affected by the event, given that an event

actually occurs. The results are summarised below. (Note: Setups 1-3d are laid out in detail in 6.2).

	Error rate (%)							
Setup ID	**Setup 1**	**Setup 2a**	**Setup 2b**	**Setup 2c**	**Setup 3a**	**Setup 3b**	**Setup 3c**	**Setup 3d**
Stage 1: Detect number of vehicles	0%	0%	0%	0%	0%	0%	0%	0%
Stage 2: Detect number of vehicles	0%	**14%**	1%	0%	0%	0%	0%	0%
Stage 2: Detect vehicle affected	NA	2%	6%	**16%**	**60%**	**41%**	**43%**	**20%**

TABLE 6.1: Error rate % over 1,000 simulations

The *large* error percentages have been highlighted in the table above. As can be seen, for stage 1 (row 1 in the table above), we get no errors at all. This tells us that the method reliably identifies the correct number of vehicles before they enter the occlusion zone. For stage 2 (row 2 in the table above), we do get some errors in number of vehicles detected in the cases when there is a vehicle *disappearance* in the occlusion zone. The errors are more pronounced for the case when the first vehicle (among the 3 vehicles) disappears and the method fails to identify that the number of vehicles has reduced.

On further analysis of these particular simulations, we find that in these cases, *before* the car disappearance event for vehicle 1 occurs inside the occlusion zone, all the 3 vehicles have managed to attain a separation distance of approx. 8m and a velocity of approx. 5m/s. This means that *before* the event both vehicles 2 and 3 are very close to their target average velocity as well as the separation distance between vehicles being close to 8m, there is very little contribution from the speeding up or braking term to their acceleration. Thus when the vehicle 1 disappearance event actually occurs, there is hardly any *change* in the velocity and acceleration of vehicles 2 and 3. Loosely speaking, vehicle 1 disappearance event does not have any substantial impact on the motion of vehicles 2 and 3. Thus the Bayes factor based model methodology fails

to detect any *change* in the motion dynamics of the observed vehicles 2 and 3 and so it assumes (incorrectly) that vehicle 1 is totally occluded (i.e. hidden from the sensor) post coming out from the occlusion zone and *no* event has occurred and hence the number of vehicles has not changed. This is what gives rise to the errors. Note that a Bayes factor based model selection technique will always prefer the model with fewer parameters/simpler complexity and in these particular simulations, it so happens that the observed data does not justify the additional complexity arising from assuming an event has occurred at a particular time point followed by a change in the motion dynamics of the vehicles. In general however, the low error rates in rows 1 and 2 in the table above show that the method performs very well in detecting the true number of vehicles.

The third row in Table 6.1 shows the error % of the ability of the method to detect which particular vehicle has appeared/disappeared due to the *event* in the occlusion zone, given that the correct number of vehicles has been identified by the method in Stage 2. The results show that the errors rates are quite small when the event is a vehicle *disappearance*, but when the event is a vehicle *appearance*, the error rates are comparatively much higher. However, note that if we were randomly guessing the identity of the new vehicle in setups 3a-3d, we would get error rates of 75% throughout. So our method definitely performs much better than random guessing.

The reason that setups 3a-3d generate higher error rates than setups 2a-2c is as follows: To accommodate an extra vehicle, we have to increase the dimension of $\mathbf{X}$ corresponding to our particle filter for a given model (see 6.4). Since we randomly initialise the x-position, velocity and acceleration of the new vehicle, with position being within a range (i.e. inside the occlusion zone and

between two given vehicles, depending on where we assume the new vehicle starts), this gives a wide range for selecting the starting position. The velocity and acceleration also have a wide range for selection of possible starting values. Thus the chosen starting values for the state space of the new vehicle may be quite different than the actual ground truth. These starting values are then propagated forward in the particle filter for this particular model. Hence it is possible that a different choice of the starting positions of the new vehicle (which is closer to the true value) but with a different model (i.e. assuming that the identity of the new vehicle is now different from the ground truth) may give a higher likelihood in a given simulation, once the measurements resume. This is what causes the errors in identifying which vehicle is affected by an event. However, to test that our method is robust to the above effect, we also looked at the second best prediction of the identity of the new vehicle from our method, where second best prediction means which vehicle is predicted by our method to be the second most likely to be affected by the event. More precisely, this means we choose the i which is the *second largest* in 6.8. Taking this second best prediction into account, the error rates in row 3 for setups 3a-3d dropped to $\in$ [32%, 2%], thus demonstrating that our method of identifying the vehicle affected by the event is not a long way off from the ground truth.

Finally, for setups 2a-2c and 3a-3b, we also get estimates of the distance at which the event occurred, as outlined in the Methodology section earlier (See 6.3). For each simulation, where we get an estimate of the distance of the event, we calculate the corresponding error in our estimate (error = estimated distance of event - true distance of event). Below we plot the histograms of these errors for the various cases examined above.

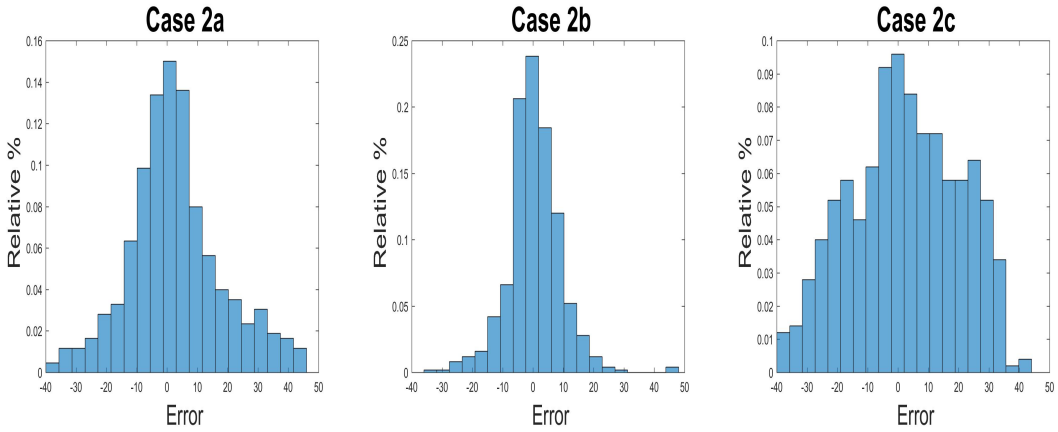

FIGURE 6.1: Histogram of errors in estimate of the distance of the event (vehicle disappearance)

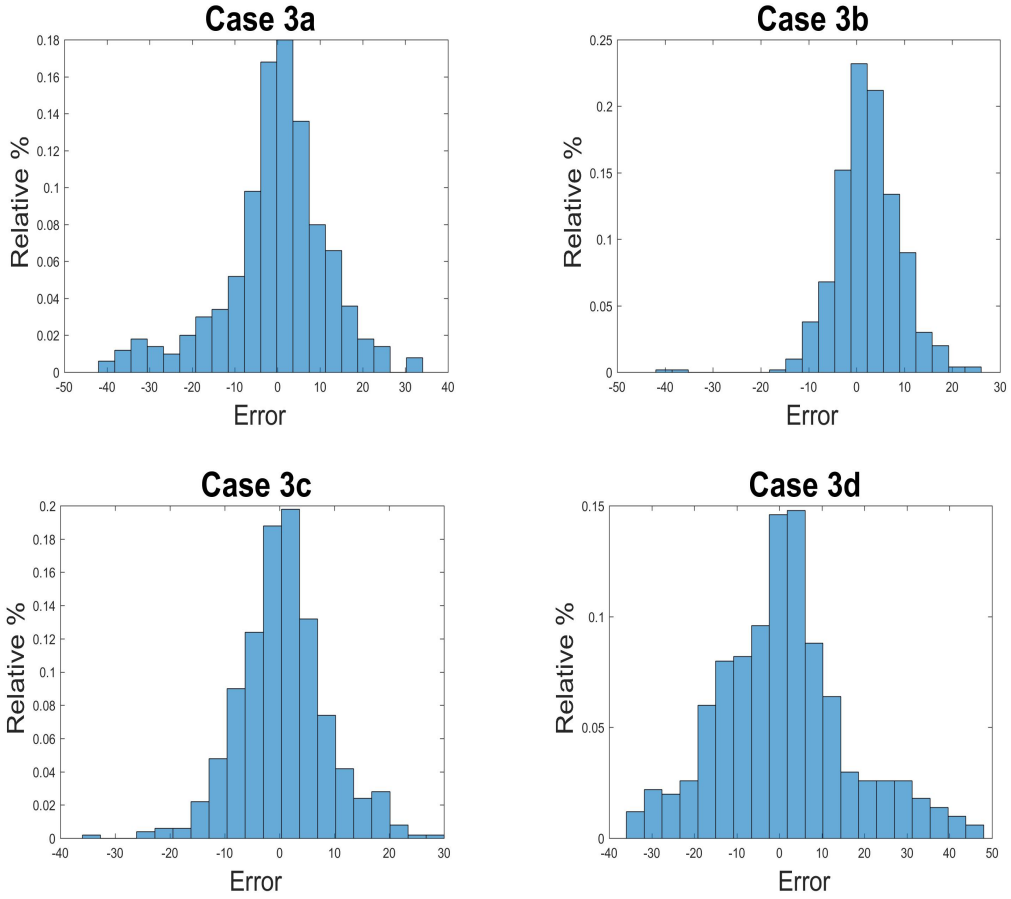

FIGURE 6.2: Histogram of errors in estimate of the distance of the event (vehicle appearance)

Note that since the occlusion zone stretches for a maximum width of 50m, the range of the errors in the distance estimates $\in$ [-50m, 50m] as can be seen in the range of the histograms above. Finally note that, had we randomly guessed the distance of the event, given that we are also generating the (ground truth) distance of the event randomly, we would have got a triangular distribution supported on [-50m, 50m] with the peak at 0 (as the difference of two uniform random variables on the same support gives a triangular distribution centered at 0). Thus the histograms above with a higher peak at 0 and more kurtosis compared to a triangular distribution show that our method performs better than random guessing in predicting the distance of the event.

Finally, note that all throughout the experimental setups above we have started with 3 vehicles as the ground truth before the occurence of an event changes the number of vehicles. To test how our method scales to higher number of vehicles, we performed one final experimental run where we start with 4 vehicles as the ground truth (followed by events) and then look at the average error rates obtained as compared to the case of starting with 3 vehicles as the ground truth. However note that, in this comparison case, since we have a higher number of starting vehicles (i.e. 4), we increase the width of our occlusion zone from [100m, 150m] to [100m, 175m], so that we ensure that there is a significant amount of time when *all* the 4 vehicles are occluded when passing through the occlusion zone. In order to make the comparison fair, we re-run the case with 3 vehicles but with the new occlusion zone of [100m, 175m]. We run 500 simulations for each sub-case of the above setups and present the summarised results below:

	Error Rate %			
	Starting Number of Vehicles	**No Event**	**Vehicle Disappearance**	**Vehicle Appearance**
Stage1: Detect number of vehicles	**3 vehicles**	0%	0%	0%
	4 vehicles	0%	0%	0%
Stage 2: Detect number of vehicles	**3 vehicles**	5%	3%	0%
	4 vehicles	1%	5%	0%
Stage 2: Detect vehicle affected	**3 vehicles**	NA	**15%**	**39%**
	4 vehicles	NA	**14%**	**49%**

TABLE 6.2: Error rate % over 500 simulations
(comparison between 3 vehicles and 4 vehicles)

The *large* errors have been highlighted in the table above. The first four rows in the table show that increasing the number of vehicles does not significantly affect the ability of our method to detect the true number of vehicles both before and after the event. However, the error rates do go up from the case of 3 starting vehicles to 4 starting vehicles (rows 5-6 in the table) when our method has to detect which particular vehicle has appeared/disappeared due to an event in the occlusion zone, given that the correct number of vehicles has been identified in Stage 2. Even then, we note that for the case with 4 starting vehicles, had we randomly guessed the identity of the vehicle affected by the event, the error rates would be $\in$ [75%, 80%] throughout. So we conclude that the quality of our method does not deteriorate substantially even if we have larger number of targets to track.

6.6 Conclusions

In this chapter we have presented results which show that even in cases of substantial occlusion where *all* the vehicles are hidden from the sensor for a significant amount of time, our method still manages to capture the true number of

vehicles in almost all cases. The only times it fails are when the vehicles have attained a relative *equilibrium* with respect to each other such that there are no significant observable *interactions* between them. In a dense urban traffic scenario, we expect such situations to arise very infrequently, and thus we expect our method to be very close to 100% accuracy in detecting the true number of vehicles.

In addition, when an event occurs (i.e. vehicles enter or leave our target group of vehicles unseen to the measurement sensor), our method is able to deduce the changed number of vehicles dynamically and thus does not lose track of the original group of vehicles it was tracking. Moreover, the method allows us to make a prediction of the actual vehicle affected, and hence this step automatically does data association, i.e. relate measurements (once they have resumed) to their corresponding vehicles once they have emerged from the occlusion zone. We have also showed that the quality of this prediction is robust to the number of vehicles we are tracking.

Finally, we also get an estimate of the distance of the event (if it occurs), as another output of our method. So if we observe many separate independent events on the same stretch of road, but the events are occurring in an occlusion zone, then using the distance estimates above, we can build up a probability distribution of where the events take place. Thus for example, when we have vehicles disappearing or new vehicles appearing (i.e. joining in) the group of vehicles we are tracking, because the vehicles are driving into and out of a car park where the entrance/exit of the car park is hidden from the sensor's field of view, the distance estimates above would allow us to make a fairly precise (i.e. low variance) estimate of where the actual entrance/exit of the car park is. Additionally, once we have got such an estimate from observing

multiple events, we can then restrict the search for $T^{*(opt)}_{N_2,i}$ in 6.6 such that we only allow events to occur in a small neighbourhood of this estimated point in the occlusion zone. This would drastically speed up the model selection process.

Chapter 7

Future Work

In this final chapter, we present some avenues for future work in this area. The items listed below either remove or relax some of the assumptions made in this project or lay out directions for further refinement of the ideas presented in the earlier chapters.

1. The first major area to develop further would be to introduce *clutter* in the experimental setup. This means that we allow for mis-detections of vehicles by our measurement sensor. Various methods are available to tackle this aspect of the problem. For example, optimal all-neighbour multi-target tracking in clutter enumerates all possible joint measurement-to-track assignments and calculates the a posteriori probabilities of each of these joint assignments. However, the numerical complexity of this process is combinatorial in the number of tracks and the number of measurements. There is also the Viterbi Data Association (VDA) algorithm, in which the target motion is assumed to be a Markov process. Various authors have explored multi-target tracking in the presence of clutter in great detail employing a variety of approaches (See [2], [10], [36] and [38]). Such methods can be explored in future to extend our work to incorporate clutter in the problem setup.

2. We have used a relatively simple vehicle dynamics model in this project as described in Chapter 3. One can explore more complex vehicle motion models simulating realistic driver behaviour in dense urban traffic scenarios. As described in 2.3 and 3.1, there is already an existing body of research exploring traffic simulation models. Such advanced models can be used to generate the ground truth to determine how well our method performs in such cases.

3. Throughout this project we have treated vehicles as *point* objects. Hence an obvious extension to this work would be to relax this assumption and treat vehicles as *extended* objects, such that each vehicle generates multiple measurements. Our method could be adapted to deal with such scenarios by, say estimating the centroid of each vehicle from the set of measurements received per vehicle and then proceed as before. More complex refinements may also be possible.

4. We have made some simplifying assumptions regarding the type of events to keep computational costs low. Our assumptions were that two events cannot occur within an single occlusion zone and in any event only one vehicle can be affected. In future work, these assumptions could be relaxed to allow multiple events affecting multiple vehicles within a single occlusion zone, at the expense of increasing computational costs.

5. In computer vision based tracking techniques, researchers have used pixel based information like colour/contour/shape to detect and track targets. In our project, all we have used is the position of the vehicle as detected by the sensor. Incorporating additional information like colour/shape into the measurement process can greatly reduce the possible number of

competing models we need to consider at each stage. For example, if we observed three red vehicles enter the occlusion zone and three red and one green vehicle emerge from the occlusion zone, it is easy to pick out which is the *new* vehicle simply based on the colour. Hence, future work could focus on incorporating such additional information into our method. In this case the measurement observations are vector valued (say position and colour) and thus we would have additional terms in the measurement likelihood expression (See 4.2).

6. Finally, note that all the work above has been done on simulated datasets. Hence the final extension of this work would be to implement our method on real life datasets and determine the performance both in terms of *quality* of tracking as well as real-time speed.

Bibliography

[1] H. Stern A. Gelman J. Carlin and D. Rubin. In: *Bayesian Data Analysis. London: Chapman & Hall* (1995).

[2] G. Ahmed and M. Farooq. "Single Target Tracking in Clutter: Performance Comparison between PDA and VDA". In: *6th international Conf. on Information Fusion* (2003).

[3] N. Ma B. T. Vo C. M. See and W. T. Ng. "Multi-Sensor Joint Detection and Tracking with the Bernoulli filter". In: *IEEE Transactions on Aerospace and Electronic Systems* 48.2 (2012), pp. 1385–1402.

[4] J. Bernardo and A. F. M. Smith. In: *Bayesian Theory. John Wiley* (1994).

[5] J. Park C. Aeschliman and A. C. Kak. "Tracking Vehicles Through Shadows and Occlusions in Wide-Area Aerial Video". In: *IEEE Transactions on Aerospace and Electronic Systems* 50.1 (2014), pp. 429–444.

[6] M. Bakich C. Yang and E. Blasch. "Nonlinear Constrained Tracking of Targets on Roads". In: *Proceedings of the 8th International Conference on Information Fusion* 1 (2005), pp. 235–242.

[7] Jonathan Chemla. "Multi-target tracking using vehicle mounted sensors Adaptation of the Kalman filter in the presence of occlusion". UCL, 2014.

[8] W.Burgard D. Fox S.Thrun and F. Dellaert. "Particle filters for mobile robot localisation". In: *Sequential Monte carlo Methods in Practice (A. Doucet, N. de Freitas, and N.J.Gordon, eds.)* (2001), pp. 927–934.

[9] V. Shvetsov D. Helbing A. Hennecke and M. Treiber. "Micro- and Macro-Simulation of Freeway Traffic". In: *Mathematical and Computer Modelling* 35.5-6 (2002), pp. 517–547.

[10] A. Gad and M. Farooq. "Tracking Highly Maneuvering Targets in Clutter using Interacting Multiple Model Fuzzy Logic Based Tracker". In: *6th international Conf. on Information Fusion* 4729 (2002).

[11] Fergyanto E Gunawan. "Two-vehicle dynamics of the car-following models on realistic driving condition". In: *Journal of Transportation Systems Engineering and Information Technology* (2012), 67 – 75.

[12] P. Hidas. "Modelling Lane Changing and Merging in Microscopic Traffic Simulation". In: *Transportation Research Part C: Emerging Technologies* 10.5-6 (2002), pp. 351–371.

[13] Y. C. Ho and R. C. K. Lee. "A Bayesian approach to problmes in stochastic estimation and control". In: *IEEE Trans. Automatic Control* 9 (1964), pp. 333–339.

[14] P. Horridge and S. Maskell. "Real-Time Tracking of Hundreds of Targets With Efficient Exact JPDAF Implementation". In: *Proceedings of the 7th International Conference on Information Fusion* (2006), pp. 1–8.

[15] M. Isard and A. Blake. "CONDENSATION - conditional density propagation for visual tracking". In: *International Journal of Computer Vision* 29.1 (1998), pp. 5–28.

[16] Zeng J. Tu Z. Zhang and T. Huang. "Face localization via hierarchical CONDENSATION with Fisher boosting feature selection". In: *CVPR 2004: Proceedings of the 2004 Computer Society Conference of Computer Vision and Pattern Recognition* (2004), pp. II–719–II–724.

[17] H. Ai J. Xing and S. Lao. "Multi-Object Tracking Through Occlusions by Local Tracklets Filtering and Global Tracklets Association with Detection Responses". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009), pp. 1200–1207.

[18] W.H. Jefferys and J.O. Berger. "Ockham's razor and Bayesian analysis". In: *American Scientist* 80 (1992), pp. 64–72.

[19] G. Kitagawa. "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models". In: *Journal of Computational and Graphical Statistics* 5.1 (1996), pp. 1–25.

[20] T.Tan L.Wang H.Ning and W.Hu. "Fusion of static and dynamic body biometrics for gait recognition". In: *IEEE Transactions on Circuits and Systems for Video Technology* 14.2 (2004), pp. 149–158.

[21] R. P. S. Mahler. "Statistical Multisource-Multitarget Information Fusion". In: (2007).

[22] D. J. Salmond N. J. Gordon and A. F. M. Smith. "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". In: *IEEE Proc.-F* 140.2 (1993), pp. 107–113.

[23] K. Palaniappan R. Pelapur and G. Seetharaman. "Robust Orientation and Appearance Adaptation for Wide-Area Large Format Video Object Tracking". In: *9th International Conference on Advanced Video and Signal-Based Surveillance (AVSS)* (2012), pp. 337–342.

[24] C. E. Rasmussen and Z. Ghahramani. "Occam's razor". In: *In Advances in Neural Information Processing Systems 13* (2001).

[25] D. B. Reid. "An Algorithm for Tracking Multiple Targets". In: *IEEE Transactions on Automatic Control* 24.6 (1979), pp. 843–854.

[26] R.Green and L. Guan. "Quantifying and recognizing human movement patterns from monocular images: Parts I and II". In: *IEEE Transactions on Circuits and Systems for Video technology* 14.2 (2004), pp. 179–198.

[27] J. Li S. K. Pang and S. J. Godsill. "Detection and Tracking of Coordinated Groups". In: *IEEE Transactions on Aerospace and Electronic Systems* 47.1 (2011), pp. 472–502.

[28] S. Russell S. Oh and S. Sastry. "Markov Chain Monte Carlo Data Association for Multiple-Target Tracking". In: *IEEE Transactions on Automatic Control* 54.3 (2009), pp. 481–497.

[29] S. Karthikeyan S. Santhoshkumar and B. Manjunath. "Robust Multiple Object Tracking by Detection with Interacting Markov Chain Monte Carlo". In: *Image Processing (ICIP), 2013 20th IEEE International Conference* (2013), pp. 2953–2957.

[30] David Salmond and Neil Gordon. *An introduction to particle filters*. 2005.

[31] M.Orten S.Arulampalam N.Gordon and B.Ristic. "A variable structure multiple model particle filter for GMTI tracking". In: *Fusion 2002: Prooceedings of the 5th international conference on Information Fusion* (2002), pp. 927–934.

[32] I. Szottka and M. Butenuth. "An Adaptive Particle Filter Method for Tracking Multiple Interacting Targets". In: *MVA2011 IAPR Conference on Machine Vision Applications* (2011), pp. 6–9.

[33] J. K. Rosenblatt T. Bailey E. M. Nebot and H. F. Durrant-Whyte. "Data Association for Mobile Robot Navigation: A Graph Theoretic Approach". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (2000), pp. 2512–2517.

[34] J. K. Uhlmann. "An Introduction to the Combinatorics of Optimal and Approximate Data Association". In: *Multisensor Data Fusion, D. L. . Hall and J. Llinas, Eds.* (2001).

[35] S. G. Fabri V. Kadirkamanathan M. H. Jaward and M. Kadirkamanathan. "Particle Filters for Recursive Model Selection in Linear and Nonlinear System Identification". In: *Proceedings of the 39th IEEE Conference on Decision and Control* (2000).

[36] K. Xing X. Yang and X. Feng. "Maneuvering Target Tracking in Dense Clutter Based on Particle Filtering". In: *Chinese Journal of Aeronautics* 24.2 (2011).

[37] T. Balch Z. Khan and F. Dellaert. "MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.11 (2005), pp. 1805–1819.

[38] B. Zhou. "Algorithms for Data Association and State Estimation". In: *Ph.D. Dissertation, Dept. of Elec. and Comp. Eng., Pennsylvania State University, PA* (1992).