

# Forensic Tracking and Surveillance: Algorithms for Homogeneous and Heterogeneous Settings

Saif Mohammed S. A. Al-Kuwari

Technical Report  
RHUL-MA-2012-2  
25 August 2011



Department of Mathematics  
Royal Holloway, University of London  
Egham, Surrey TW20 0EX, England

<http://www.rhul.ac.uk/mathematics/techreports>

Forensic Tracking and Surveillance  
Algorithms for Homogeneous and Heterogeneous Settings

Submitted by

Saif Mohammed S. A. Al-Kuwari

for the degree of Doctor of Philosophy

of

Royal Holloway, University of London

2011

### **Declaration**

I, Saif Mohammed S. A. Al-Kuwari, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed ..... (Saif Mohammed S. A. Al-Kuwari)

Date:

و ما توفيقى إلا بالله

*“I cannot succeed except through God”*

## Abstract

Digital forensics is an emerging field that has uniquely brought together academics, practitioners and law enforcement. Research in this area was inspired by the numerous challenges posed by the increased sophistication of criminal tools. Traditionally, digital forensics has been confined to the extraction of digital evidence from electronic devices. This direct extraction of digital evidence, however, no longer suffices. Indeed, extracting completely raw data without further processing and/or filtering is, in some cases, useless. These problems can be tackled by the so-called “computational forensics” where the reconstructs evidence are undertaken further processing. One important application of computational forensics is criminal tracking, which we collectively call “forensic tracking” and is the main subject of this thesis. This thesis adopts an algorithmic approach to investigate the feasibility of conducting forensic tracking in various environments and settings. Unlike conventional tracking, forensic tracking has to be passive such that the target (who is usually a suspect) should not be aware of the tracking process. We begin by adopting pedestrian setting and propose several online (real-time) forensic tracking algorithms to track a single or multiple targets passively. Beside the core tracking algorithms, we also propose other auxiliary algorithms to improve the robustness and resilience of tracking. We then extend the scope and consider vehicular forensic tracking, where we investigate both online and offline tracking. In online vehicular tracking, we also propose algorithms for motion prediction to estimate the near future movement of target vehicles. Offline vehicular tracking, on the other hand, entails the post-hoc extraction and probabilistic reconstruction of vehicular traces, which we adopt Bayesian approach for. Finally, the contributions of the thesis concludes with building an algorithmic solution for multi-modal tracking, which is a mixed environment combining both pedestrian and vehicular settings.

## Acknowledgement

First and foremost, all thanks and praised are due to God, for giving me the strength and patience to complete this thesis, like many other things in life. This project (and everything I have ever done) could not have been completed without his blessings.

On Earth, I would like to thank my supervisor, Dr. Stephen D. Wolthusen, for his support and guidance throughout my PhD, he not only taught me how to be an academic researcher, but also how to behave like one, being flexible, patient and persistent. I would also like to thank my external examiner Dr. Frank Kargl of University of Twente and my internal examiner Dr. Geraint Price of Royal Holloway for their insightful comments on my thesis and for making my viva an interesting experience.

During my stay in Royal Holloway, I was privileged to have met many great people. I would like to thank them all and wish them well in their life and careers. I am particularly in debt for Ahmed Al-Mulla for forcing me to join him to a trip to London and for Hezam Abdulla for calling us on our way and making us divert to Egham, where I met Ziyad Al-Salloum who convinced me to join the Information Security Group (ISG) in Royal Holloway. If any of these people was missing that day, I would not have known Royal Holloway and this thesis probably would not have been written, who said life is predictable? At Royal Holloway, I am grateful for the unique research experience I had. Particularly, I would like to thank Dr. Allan Tomlinson, Prof. Peter Wild, Prof. Keith Martin, Prof. Chris Mitchell, Dr. Carlos Cid and Prof. Kenny Paterson.

Back home, I would like to thank my two sponsors, Qatar's Higher Education Institute (HEI) and Ministry of Foreign Affairs (MOFA) for their financial support over the years. From HEI, I would like to especially thank Aisha Al-Motawa, for being such a great academic advisor throughout the first two years of my scholarship, and Eid Al-Hajri for being the same in the last one. I would also like to thank Hind Al-Swaidi and Fatma Al-Saadi for their support. From MOFA, I would like to thank Reem Al-Derham for managing my scholarship at her end. Special thanks go to Mohammed Al-Kaabi, Qatar's cultural attaché in London for his help and support since 2002, when I first came to the UK for my undergraduate.

Last but not least, I would like to thank my family for always believing in me, even more than I believed in myself. Thanks to my mother for putting up with all the holidays I had to miss during the course of my PhD and for accepting all my artificial excuses. Thanks to my father for his inspiration, and for being such a great role model, I am not sure I will ever achieve what he has achieved, but I can promise I will do my best. Thanks for my sisters and brothers for being there for me. Thanks for the endless prayers of my grandmother, who never got tired from asking me the same question over and over again, when are you coming home? Now I can finally answer her, soon.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Contributions and Publications . . . . .	3
1.2	Thesis Outline . . . . .	5
<b>I</b>	<b>Forensic Tracking and Localisation</b>	<b>8</b>
<b>2</b>	<b>Localisation Techniques</b>	<b>9</b>
2.1	Introduction . . . . .	9
2.2	Localisation in Sensor Networks . . . . .	11
2.2.1	Parameter Measurement . . . . .	12
2.2.2	Geometric Location Estimation . . . . .	13
2.3	Localisation in Cellular Networks . . . . .	17
2.4	Localisation Fusion . . . . .	19
2.4.1	Fusing Different Technologies . . . . .	19
2.4.2	Fusing Different Parameters . . . . .	20
2.5	Accuracy Issues . . . . .	21
2.6	Summary . . . . .	22
<b>3</b>	<b>Forensic Tracking and Mobility Models</b>	<b>23</b>
3.1	Introduction . . . . .	23
3.2	Online Tracking . . . . .	24
3.2.1	Active Tracking . . . . .	24
3.2.2	Passive tracking . . . . .	26
3.2.3	Reliability and Security . . . . .	26
3.2.4	Privacy Implications . . . . .	27
3.3	Offline Forensic Tracking . . . . .	28
3.3.1	Basic Bayesian Approach . . . . .	28
3.3.2	Dynamic Bayesian Networks . . . . .	29

3.4	Mobility Models . . . . .	30
3.4.1	Pedestrian Mobility . . . . .	31
3.4.2	Vehicular Mobility . . . . .	33
3.4.3	Multi-modal Mobility . . . . .	36
3.5	Summary . . . . .	37
<b>II</b>	<b>Online Forensic Tracking</b>	<b>38</b>
<b>4</b>	<b>Online Pedestrian Forensic Tracking</b>	<b>39</b>
4.1	Introduction . . . . .	39
4.2	Scene Setting and Assumptions . . . . .	41
4.3	Online Pedestrian Localisation . . . . .	44
4.3.1	Passive Localisation . . . . .	45
4.3.2	Active Localisation . . . . .	48
4.4	Piconets . . . . .	49
4.4.1	Tracking Piconet . . . . .	49
4.4.2	Connecting Piconet . . . . .	50
4.5	Piconet Formation . . . . .	50
4.5.1	Formation by Direct Interrogation (FDI) . . . . .	50
4.5.2	Formation by Neighbour Interrogation (FNI) . . . . .	52
4.6	Basic Pedestrian Tracking . . . . .	53
4.6.1	Agent Recruitment and Retirement . . . . .	53
4.6.2	Simulation Results . . . . .	54
4.7	Advanced Pedestrian Tracking . . . . .	55
4.7.1	Tracking with Multiple Trackers . . . . .	57
4.7.2	Tracking Multiple Targets . . . . .	60
4.7.3	Fault Tolerance . . . . .	60
4.7.4	Leader Election . . . . .	63
4.7.5	Transmission Algorithm . . . . .	64
4.7.6	Simulation Results . . . . .	66
4.8	Privacy in Online Forensic Tracking . . . . .	68
4.9	Summary . . . . .	69
<b>5</b>	<b>Vehicular Forensic Tracking and Motion Prediction</b>	<b>70</b>
5.1	Introduction . . . . .	70
5.2	Vehicular Networks . . . . .	72
5.3	Vehicular Localisation . . . . .	73

5.4	Mobility Prediction . . . . .	74
5.4.1	Time Prediction . . . . .	75
5.4.2	Direction Prediction . . . . .	77
5.5	Vehicular Tracking . . . . .	80
5.6	Simulation Results . . . . .	82
5.7	Vehicular Parameter Estimation . . . . .	85
5.8	Summary . . . . .	87
<b>III</b>	<b>Offline Forensic Tracking</b>	<b>88</b>
<b>6</b>	<b>Bayesian Offline Vehicular Forensic Tracking</b>	<b>89</b>
6.1	Introduction . . . . .	89
6.2	Trace Fusion . . . . .	91
6.3	Trace Reconstruction . . . . .	94
6.3.1	Phase 1: Routes Identification . . . . .	95
6.3.2	Phase 2: Routes Analysis and Selection . . . . .	99
6.4	Simulation Results . . . . .	102
6.5	Offline Estimation Accuracy . . . . .	105
6.6	Summary . . . . .	106
<b>7</b>	<b>Offline Multi-modal Forensic Tracking</b>	<b>107</b>
7.1	Introduction . . . . .	107
7.2	Trace Reconstruction Framework . . . . .	109
7.3	Scene Representation . . . . .	111
7.4	Mobility Modelling . . . . .	117
7.4.1	Pedestrian Mobility Delay Model . . . . .	118
7.4.2	Transport Mobility Delay Models . . . . .	120
7.4.3	Multi-modal Mobility Delay Model . . . . .	121
7.5	Fuzzy Trace Validation . . . . .	124
7.5.1	Fuzzy Logic . . . . .	125
7.5.2	Fuzzification . . . . .	126
7.5.3	Fuzzy Inference . . . . .	126
7.5.4	Defuzzification . . . . .	127
7.6	Trace Reconstruction . . . . .	128
7.6.1	The Algorithm . . . . .	130
7.6.2	Complexity Analysis . . . . .	134
7.7	Summary . . . . .	137

<b>8 Conclusion and Future Work</b>	<b>138</b>
8.1 Offline Pedestrian Forensic Tracking . . . . .	139
8.2 Advanced Bayesian-based Trace Reconstruction . . . . .	139
8.3 Tracking based on Social Networking . . . . .	140
8.4 Crime Reconstruction . . . . .	140
8.5 Multi-modal Trace Reconstruction System . . . . .	141
8.6 Live Vehicular Forensics . . . . .	142
8.7 Final Remarks . . . . .	142
<b>Bibliography</b>	<b>142</b>

# List of Figures

1.1	Dependancies among the chapters of the thesis . . . . .	7
2.1	Triangulation . . . . .	14
2.2	Advanced Trilateration . . . . .	16
3.1	Sample Dynamic Bayesian Network (DBN) . . . . .	30
3.2	Forces determining pedestrian mobility . . . . .	33
3.3	Scenario illustrating our sample vehicular mobility model . . . . .	36
4.1	A sample single-target single-tracker tracking network consisting of one tracking piconet and one connecting piconet . . . . .	43
4.2	Basic Trilateration . . . . .	47
4.3	Flow chart illustrating the FDI and the FNI algorithms . . . . .	51
4.4	Difference in meters between the actual and estimated target locations during 30 simulation minutes . . . . .	56
4.5	Mutual Authentication . . . . .	58
4.6	Virtual Tracking . . . . .	61
4.7	Maximum and minimum number of agents forming the piconets . . . . .	63
4.8	The Transmission Algorithm (TA) . . . . .	66
4.9	Simulation results when adopting the Random Waypoint model . . . . .	67
4.10	Simulation results when adopting the Brownian Walk model . . . . .	67
4.11	Simulation results when adopting the Gauss-Markov model . . . . .	68
5.1	Illustration of how the speed of a vehicle is estimated in the time predi- cation algorithm . . . . .	76
5.2	Intersection regulatory rules . . . . .	78
5.3	Simulation results for 10 nodes scenario . . . . .	85
5.4	Simulation results for 30 nodes scenario . . . . .	86
5.5	Simulation results for 50 nodes scenario . . . . .	86
5.6	Simulation results for 100 nodes scenario . . . . .	87

6.1	Trace fusion of RF and visual based tracks . . . . .	94
6.2	Sample trace with missing data . . . . .	96
6.3	Possible routes through a sample gap . . . . .	97
6.4	Simulation Results . . . . .	103
7.1	Sample target trace (with gaps) . . . . .	109
7.2	Illustration of the scene preparation phase . . . . .	112
7.3	A graph of three routes $R_1, R_2, R_3$ . . . . .	116
7.4	Illustration of the manoeuver behaviour of a target and how to calculate the extra distance . . . . .	119
7.5	Illustration of the 3D matrix $M^{v_i}$ . . . . .	124
7.6	Fuzzy Inference System . . . . .	126
7.7	Input and output variables plot . . . . .	127
7.8	Graphical representation of the relationship between the input/output variables of our fuzzy system . . . . .	129

# List of Abbreviations

AOA	Angle of Arrival
BRC*	Bounded Route Counter (algorithm)
CCTV	Closed Circuit Television
(D)BN	(Dynamic) Bayesian Network
EDR	Event Data Recorder
FDI*	Formation by Direction Interrogation (algorithm)
FIS	Fuzzy Inference System
FNI*	Formation by Neighbour Interrogation (algorithm)
GPS	Global Positioning System
IDM	Intelligent Driver Motion (mobility model)
IVC	Inter Vehicle Communication
MANET	Mobile Ad hoc Network
MMDM*	Multi-modal Mobility Delay Model
NS-2	Network Simulator 2
NTMDM*	Non-traffic-based Transport Mobility Delay Model
PMDM*	Pedestrian Mobility Delay Model
POI	Point of Interest
RF	Radio Frequency
RIPA	Regulation of Investigatory Power Act 2000
RSS(I)	Received Signal Strength (Indication)
SH*	Secure Handover (algorithm)
T-SW*	Tracking Software
TA*	Transmission Algorithm
TDOA	Time Difference of Arrival
TOA	Time of Arrival
TTMDM*	Traffic-based Transport Delay Mobility Model
VAID*	Virtual Agent ID
VANET	Vehicular Ad hoc Network
VPID*	Virtual Piconet ID
VTN*	Virtual Tracking Network
WBS*	Weighted Bound Search (algorithm)
WSN	Wireless Sensor Network

\*novel

# List of Algorithms

4.1	Single Target Localisation	48
4.2	Round-trip Time	49
4.3	Formation by Direct Interrogation (FDI)	52
4.4	Formation by Neighbour Interrogation (FNI)	53
4.5	Detection of a Departing Salve	55
4.6	Secure Handover (SH)	58
4.7	Multi-Trackers Multi-Targets Tracking	62
4.8	Detect Tracker Procedure	62
4.9	Temporary Tracker Election	64
4.10	Transmission Algorithm (TA)	65
5.1	Vehicular Localisation	74
5.2	Online Vehicular Tracking	82
5.3	Temporary Tracking	83
6.1	Trace Fusion	95
6.2	Range Expansion Procedure	95
6.3	Bounded Route Counter (BRC)	98
6.4	Search Vertices Marking Procedure	98
7.1	Trace Reconstruction Framework	110
7.2	Gap Filling Procedure	110
7.3	Scene Representation	113
7.4	Route Marking Procedure	114
7.5	Vertex/Edge Labelling Procedure	115
7.6	End Vertices Procedure	117
7.7	Additional Edges Formation Procedure	118
7.8	Weight-Bound-Search (WBS)	133
7.9	Back-up Procedure	134

# Chapter 1

## Introduction

Alongside the advances that today's technologies brought to us, there comes a whole new class of criminal threats jeopardising our safety and security. The ubiquitous recent revolution in technology that we are witnessing today has indeed improved our life in many ways, but it also opened new doors for criminals and offenders by inadvertently creating new and more convenient means for them to conduct their criminal activities. Computer crime, cybercrime, cyberwarfare and cyber-terrorism are all terms which only came to existence a few decades ago and roughly refer to the same concept "crimes undertaken using digitalised means". As dangerous as it sounds, the technologies that gave such opportunities for criminals, are fortunately also giving us tools to fight them. Recent crime investigation techniques provided a potential framework in which law enforcement can efficiently conduct their criminal investigations. In conventional crimes, investigations are mainly based on forensic analysis of evidence left in the crime scene; such evidence were mainly biological or physical substances. These investigations are not only important to bring criminals to justice, but also help law enforcement preventing future crimes. However, sine modern crimes can be conducted in (or through) the cyberspace, conventional forensic procedures are no longer sufficient, and are largely being complemented by the so-called digital forensics.

Digital forensics is commonly conceived as the process of investigating and extracting digital evidence (traces) from electronic devices.

Traditionally, digital forensics was almost always conceded with the investigation of the *digital activities* of suspect individuals who may have used some computerised means to carry out such activities, these digital evidence may include internet browsing history, creating/editing files, downloading images/videos etc. Nonetheless, we argue that investigating the *digitalised physical activities* of those suspects is equally important. Such activities include any physical action undertaken by the suspects and

captured by digital devices. In this thesis, we will be concerned with a specific class of such activities, namely suspects' mobility behaviours. We claim that tracking a particular suspect may, in most cases, be necessary in criminal investigations; indeed, proving the location of a suspect at a particular time (or during a particular period of time) can be a significant piece of evidence in its own right. We call this class of tracking *forensic tracking*, which is basically conducting a tracking procedure for forensic purposes, and can be either online or offline. Online forensic tracking is the tracking of a suspect in real time with the added requirement of passivity, where the target is not aware of the tracking process, for obvious reasons. Offline forensic tracking, on the other hand, is actually a trace<sup>1</sup> reconstruction process, where the locations of the suspects before, during and after the crime are investigated and reconstructed if necessary. Indeed, there are scenarios where knowing where the suspects were before and after a crime is as important as (or even more important than) knowing where they were at the time of the crime. Similarly, in other scenarios knowing that the target was in a particular place away from the crime scene during the crime may be of great significance too. A prime example is a terrorist planted bomb, in which case we know that the suspects were not in the crime scene during the crime, but instead we would like to learn about their locations before and after the crime, which may help supporting some hypothesis that investigators would make.

The intelligence in the various digital forensic procedures and tools is mostly focused on searching for and extracting evidence from some digital mean. These procedures, once believed to be trivial, are becoming increasingly difficult due to the sheer data volume problem. That is, even the low profile crimes today would involved several laptops, a couple of mobile phones and perhaps one or two PC's leading to at least several 10s of Gigabytes of digital information. Clearly, forensically searching this amount of data is extremely tedious and time consuming even for the most advanced digital forensics tools; in this sense, digital forensics is reduced to information retrieval. Another challenge stems from the fact that these extraction techniques and tools are, in most cases, platform-dependant and given the numerous platforms we have nowadays, it is highly unlikely that we will ever get a universal tool that would be compatible with all platforms. However, while finding and extracting evidence is certainly non-trivial, even after extracting such evidence, in some cases the extracted *raw* evidence is totally unintelligible or severely fragmented to the point where it becomes invalid or useless. In this case, a further processing and filtering is required to forensically recover and reconstruct this data, which will allow to correlate them in a sound manner. The use

---

<sup>1</sup>Trace in this context means a physical location trace, not to be confused with biological traces such as blood stains or pieces of hair that are usually left behind in a crime scene.

of such post-processing in a forensic context is called *computational forensics* [42].

Computational forensics is the science of forensically investigating digital devices and utilise the various computational, mathematical and statistical methods to model, simulate, analyse and reconstruct pieces of evidence extracted from these devices.

Indeed, one of the main contributions of this thesis is the algorithmic treatment of a class of problems that can be considered a form of computation forensics, namely the target trace reconstruction. In trace reconstruction, we are given a tracking trace of a particular target (or a number of targets) that exhibits several tracking gaps corresponding to periods of time where tracking of the target was not possible, we are then confronted with the problem of reconstructing these tracking gaps to obtain the full target trace. In this case, if we extracted the raw target trace without further computational analysis (as the case in the traditional digital forensics procedures), we will not be able to use it as evidence because the fact that it is incomplete may render it unusable. Nevertheless, using the various computational methods, we may be able to (probabilistically) augment the missing data and obtain a full usable trace that can conceivably be used at court.

## 1.1 Contributions and Publications

This thesis contributes a collection of novel tracking algorithms covering both online and offline forensic tracking. The thesis is structured to flow logically from pedestrian forensic tracking, to vehicular forensic tracking and finally concludes with an extended discussion on multi-modal forensic tracking (which is a mixed environment comprising both pedestrian and vehicular settings). We consider various types of tracking environments and model the interaction between different mobility behaviours when tracking both single and multiple targets. While we discuss and propose algorithms for both online and offline tracking, we believe that the main contribution of this thesis is in the offline tracking scenarios since these are increasingly common in (the practical) law enforcement applications due to the unpredictability of crimes and the inherent difficulty of conducting online tracking. This thesis is based on 8 papers as follows (these form the basis of the chapters of the thesis as we discuss below):

1. S. Al-Kuwari, S. D. Wolthusen. A Survey of Forensic Localization and Tracking Mechanisms in Short-Range and Cellular Networks. In *Proc. of the 1st International Conference on Digital Forensics and Cybercrime (ICDF2C '09)*. (NY, USA, Oct. 2009), S. Goe, Ed., vol. 31 of LNICST, Springer-Verlag, pp. 19–32.

2. S. Al-Kuwari, S. D. Wolthusen. Passive Ad-Hoc Localization and Tracking in Short-Range Communication. In *Proc. of the 1st Inter. Conference on Next Generation Wireless Systems (NGWS '09)*. (Melbourne, Australia, Oct. 2009), N. Chilamkurti, Ed., ISBN 3838353463, LAB Lambert Academic Publisher.
3. S. Al-Kuwari, S. D. Wolthusen. Algorithmic Approach for Clandestine Localisation and Tracking in Short-Range Environments. In the *International Journal of Communication Networks and Distributed Systems (IJCND)* (2011). (To appear in vol. 7, no. 3 of IJCND).
4. S. Al-Kuwari, S. D. Wolthusen. Algorithms for Advanced Clandestine Tracking in Short-Range Ad Hoc Networks. In *Proc. of the 2nd International ICST Conference (MobiSec '10)*. (Catania, Sicily, Italy, May 2010), A. U. Schmidt, G. Russello, A. Liyo, N. R. Prasad, and S. Lian, Eds., vol. 46 of LNICST, Springer-Verlag, pp. 67–79.
5. S. Al-Kuwari, S. D. Wolthusen. Forensic Tracking and Mobility Prediction in Vehicular Networks. In *Advances in Digital Forensics VI, Proc. of the Sixth Annual IFIP WG 11.9 International Conference on Digital Forensics*. (Hong Kong, China, Jan. 2010), K.-P. Chow and S. Sheno, Eds., vol. 337 of IFIP Advances in Digital Forensics, Springer-Verlag, pp. 91–105.
6. S. Al-Kuwari, S. D. Wolthusen. Probabilistic Vehicular Trace Reconstruction Based on RF-Visual Data Fusion. In *Proc. of the 11th Joint IFIP TC6 and TC11 Conference on Communications and Multimedia Security (CMS '10)*. (Linz, Austria, May 2010), B. D. Decker and I. Schaumüller-Bichl, Eds., vol. 6109 of LNCS, Springer-Verlag, pp. 16–27.
7. S. Al-Kuwari, S. D. Wolthusen. Fuzzy Trace Validation: Toward an Offline Forensic Tracking Framework. In *Proc. of the 6th IEEE Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE '11)* (Oakland, CA, USA, May 2011), IEEE Press. (To appear).
8. S. Al-Kuwari, S. D. Wolthusen. Multi-Modal Trace Reconstruction: an Offline Forensic Tracking Approach. In *Proc. of the 8th Annual IFIP WG 11.9 International Conference on Digital Forensics*, Pretoria, South Africa, January, 2012. (To appear).

These papers potentially cover most scenarios in which forensic tracking can be carried out, while considering both single and multiple targets settings. Paper [1] is

the basis of parts of chapter 2 and chapter 3. Papers [2-4] are the basis of chapter 4. Paper [5] is the basis of chapter 5. Paper [6] is the basis of chapter 6. Paper [7] is the basis of parts of chapter 3 and parts of chapter 7. Paper [8] is the basis of most of chapter 7. In chapter 8 we discuss other settings and scenarios (as well as other related research topics) which can potentially extend the work of this thesis but have not been considered in the papers.

## 1.2 Thesis Outline

The flow of the thesis logically covers most of the possible scenarios and environment in which forensic tracking can be conducted. It starts with pedestrian forensic tracking, then proceeds to vehicular forensic tracking and finally concludes with multi-modal forensic tracking (which combines both pedestrian and vehicular settings). Technically, the thesis is organised in three parts. Part I consists of chapters 2 and 3 and provides a general background information which the rest of the thesis uses extensively. Part II consists of chapters 4 and 5, and considers online forensic tracking in different settings. Part III, consists of chapters 6 and 7, and considers offline forensic tracking. Parts II and III provide a collection of novel algorithms, which constitute our contributions of this thesis. Below we provide a brief summary of each chapter.

- **Chapter 1: Introduction.** This chapter provides a general motivation and overview of the main problem that this thesis is discussing, that of forensic tracking. It also summarises the contributions of the thesis and lists the publications forming the basis of this thesis. Finally, this chapter provides an outline of the rest of the thesis along with brief summary of each chapter.
- **Part I: Forensic Tracking**
  - **Chapter 2: Localisation Techniques.** In this chapter we provide a taxonomy and general overview of the most popular existing localisation techniques in mobile wireless networks, with emphases on the techniques that are commonly used in sensor and cellular networks. We also discuss ways in which different localisation techniques can be fused by adopting a multi-sensor data fusion approach, which significantly improves the localisation accuracy.
  - **Chapter 3: Forensic Tracking and Mobility Models.** This chapter builds on the techniques presented in chapter 2 and uses them in a tracking setting. Generally, tracking can either be online or offline. We discuss, in details, both online and offline tracking, and provide example applications. We also introduce the “forensic tracking” class and briefly discuss its applications. In

the second part of this chapter, we provide an algorithmic discussion about mobility models and show how they are being built in different environments; variants of these will later be used in forensic tracking settings.

- **Part II: Online Forensic Tracking**

- Chapter 4: Online Pedestrian Forensic Tracking. In this chapter we propose a number of algorithms forming a full passive forensic tracking system that can be used to track and surveil suspects and criminals using several mobile ad hoc techniques to dynamically recruit and retire tracking agents that are triggered to construct dynamically adaptable tracking networks. While developing the algorithms, we considered scenarios and settings where a single or multiple trackers track a single or multiple targets. We also propose several auxiliary algorithms to improve the security, passivity and efficiency throughout the tracking process while maintaining the tracking network.
- Chapter 5: Vehicular Forensic Tracking and Motion Prediction. After discussing passive online pedestrian tracking in chapter 4, in this chapter we proceed by considering passive online tracking in vehicular setting based on ad hoc vehicular networks (VANETs) while still adopting an agent-based tracking approach. In addition, we also propose algorithms to predict the near future movement of target vehicles and show how these predictions may greatly assist the tracking process by enabling a smoother and more reliable dynamic recruitment and retirements of the tracking agents.

- **Part III: Offline Forensic Tracking**

- Chapter 6: Bayesian Offline Vehicular Forensic Tracking. Online tracking is applicable in some scenarios, but due to the uncertainty of crimes, offline tracking may be the only plausible option in most cases. In this chapter we propose a probabilistic approach for conducting offline forensic tracking in vehicular setting, where we obtain a trace of the target vehicle and try to reconstruct it using a Bayesian statistical approach. We propose several algorithms to prepare, process and reconstruct a target's trace.
- Chapter 7: Offline Multi-modal Forensic Tracking. This chapter presents our main contribution where we combine scenarios from chapters 4, 5 and 6 to consider the problem of reconstructing traces collected at a multi-modal environment, where a target switches between different transportation modes. This is indeed a challenging environment, but more common today given the rapid advances in transport systems. While building our multi-modal

forensic tracking system, we use mobility models and transition procedures to model the transition between different mobility models. Using these techniques and other public information about the tracking scene, we show how to reconstruct the target’s full trace.

- **Chapter 8: Conclusion and Future Work.** In this chapter we conclude the thesis by providing final remarks and discussing a few possible extensions. We believe that there is a plethora of research still to be undertaken in this area.

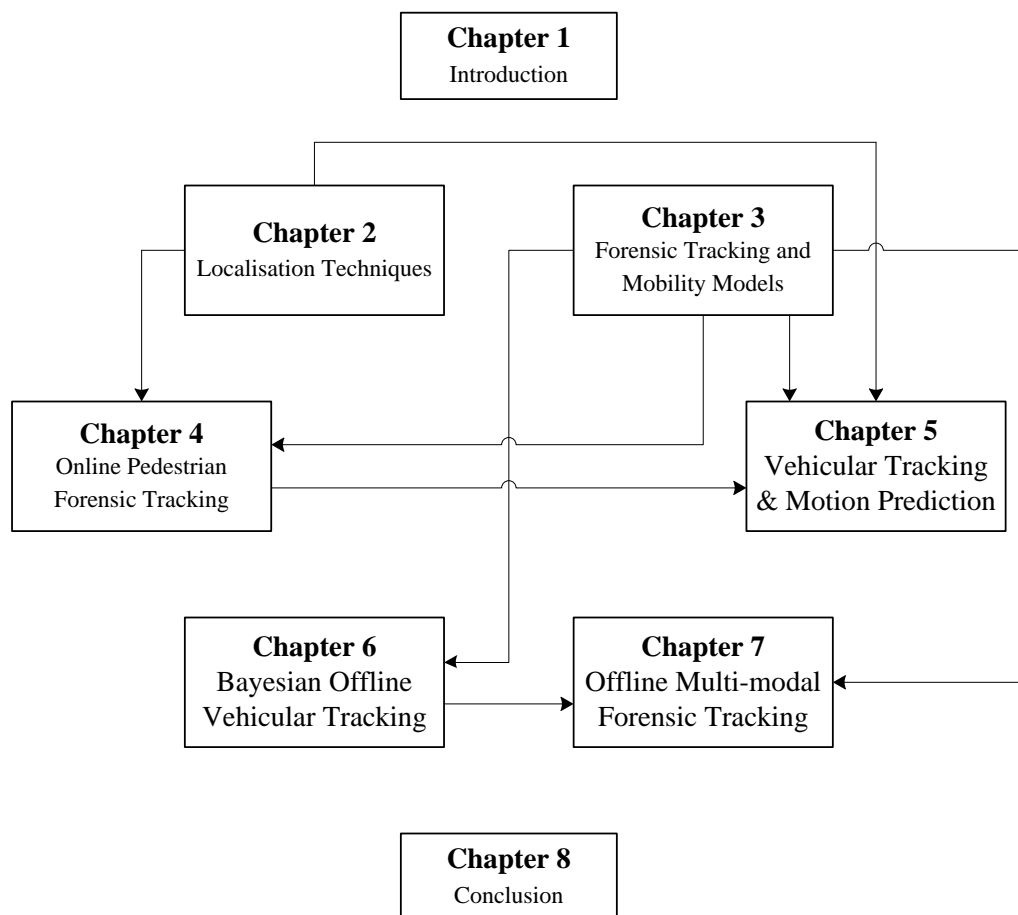


Figure 1.1: Dependancies among the chapters of the thesis

Although most of these chapters present independent results, some of them exhibit close relations, especially chapters 4 and 5 (which discuss online forensic tracking), and chapters 6 and 7 (which discuss offline forensic tracking). These relations are depicted in figure 1.1, where an arrow from chapter X to chapter Y means that material from chapter X were used/extended while developing the results/algorithms in chapter Y.

## Part I

# Forensic Tracking and Localisation

## Chapter 2

# Localisation Techniques

*Target localisation is a critical tool in criminal and, increasingly, forensic investigations. These applications are greatly aided by the proliferation of wireless technologies, most of which enable pinpointing target entities relatively efficiently. In this chapter, we provide a thorough discussion and taxonomy of both established and novel localisation techniques at different environments, we also discuss the underlying assumptions and limitations in each case. In particular, we describe the various localisation techniques for cellular, sensor, and personal area networks in both infrastructure and ad hoc environments. As individual localisation methods do not always yield the desired precision and accuracy, may require collaboration, or will be faced with excessive noise in densely built-up or highly active radio frequency environments, we additionally discuss selected approaches derived from multi-sensor data fusion applications to enhance the estimation parameters accuracy by fusing several estimation and measurement techniques. The contents of this chapter was published in [1].*

### 2.1 Introduction

Given its numerous civilian and military applications, localisation of static and mobile objects has long been an active area of research. Generally, locating an object (i.e., radio emitting device representing an individual) can either be object-based or network-based. While in the former case, the target object localises itself, in the latter, the surrounding (reference) objects localise the target object. Network-based localisation can further be active, where the reference objects collaborate with the target object to localise it, or passive, where the reference objects clandestinely localise the target

object by observing and measuring its emissions (object-based localisation is obviously active). Civilian and some military applications are usually based on active localisation where the target object is aware of the localisation process (sometimes even initiates it). On the other hand, most law enforcement applications are usually passive since the localised objects are often suspect individuals that are being investigated as part of a criminal investigation. Clearly, this passivity requirement introduces more challenges and imposes more restrictions on the localisation process as in this case it not only needs to be carried out efficiently and accurately, but also clandestinely, which is, evidently, non-trivial. In this chapter, we review various existing localisation techniques considering different environments and discuss their applications.

**Taxonomy.** Hightower *et al.* [61, 62, 60] carried out a series of related studies surveying some of the most fundamental concepts in wireless networks localisation, such as triangulation, trilateration, location proximity and scene analysis. The authors classified *Lateration* (localisation by distances) and *Angulation* (localisation by angles) as types of *Triangulation*. However, this classification is now almost obsolete restricting triangulation to angulation only and renaming lateration to *Trilateration*. The authors also discussed location proximity and scene analysis techniques. In location proximity a target object is detected as it is approaching a known reference point. On the other hand, in scene analysis, features of the surrounding environments are simplified, analysed and then compared against a predefined dataset of the characteristics of that location. Similarly, Pandy *et al.* [102] proposed a thorough classification model for localisation techniques and showed how some real localisation systems can fit in their model. The authors developed their classification based on factors influencing the localisation accuracy, including: area of deployment, physical layer technology, measurement parameters, type of lookup table (mapping measurements to locations), estimation technique, localisation entity and security parameters. For example, a localisation algorithm in sensor network can be classified as: ad hoc based (area of deployment), RF based (physical layer technology), RSS based (measurement parameter), Free Space model based (type of lookup table), probabilistic based (estimation technique), network based (localisation entity) and non-secure (security parameters).

**Localisation.** Target localisation has been an active research area and consequently the literature contains many contributions of novel and practical localisation techniques proposed for different environments and settings. As discussed in [120], localisation is generally a 2-phase process. In the first phase, parameters, such as RSS (see section 2.2.1), are estimated, and then, a geometric (triangulation/trilateration) or statistical

approach is adopted to estimate the actual location of the target. Statistical estimation methods can be parametric such as Bayesian and Maximum Likelihood (ML), or nonparametric such as k-NN (k-Nearest-Neighbor), SVR (Support Vector Regression) and neural networks. In a mobile environment (where the nodes are freely or restrictively move over an area), the mobility behaviour adds additional challenges for the localisation process. Realising this problem, Hu and Evans proposed a monte carlo localisation algorithm that actually exploit the mobility of the nodes to improve the accuracy estimation [66].

**Chapter Outline.** This chapter is organised as follows. Localisation in sensor and cellular networks are discussed in sections 2.2 and 2.3, respectively. Multi-sensor data fusion has been extensively used to improve the accuracy of the localisation process by fusing more than one localisation technique; we introduce some of these fusion approaches in section 2.4. Finally, in section 2.5, we, very briefly, discuss the affect of radio propagation on the accuracy of most localisation processes.

## 2.2 Localisation in Sensor Networks

In sensor networks, the nodes can directly communicate with each other in an ad hoc manner allowing for the adoption of less constrained algorithmic approach when designing localisation algorithms. Regardless of whether localisation is object-based or network-based, nodes localise themselves in reference to other reference nodes, this is sometimes called *self-localisation* [96], which is a standard approach in most wireless networks applications. Reference nodes have known locations and are usually located close to the node being localised, but in some applications this does not have to be the case; in fact, in GPS, the reference points are satellites (see section 2.3).

There are several ways in which a target object can be localised in reference to other objects with known locations [92]. In 2 dimensions, object  $T$  can be localised geometrically if the distance and/or angles between  $T$  and some reference objects can be accurately measured. In particular,  $T$  can be localised in two steps:

- measure distances/angles between  $T$  and other known reference objects, then
- apply a geometric procedure to determine the location of  $T$ .

In the following subsections, we first discuss the various distance/angular measurement techniques and then describe a few geometric location estimation methods; statistical localisation approaches, as discussed briefly in section 2.1, are computationally demanding and thus not discussed, see [110].

### 2.2.1 Parameter Measurement

In this section we discuss a few popular techniques for measuring the distance/angle between two or more objects, typically, single transmitter and a single or multiple receivers. These techniques assume wireless environment.

**Received Signal Strength (RSS).** The distance between a transmitter and a receiver can be estimated by measuring the strength of the transmitter's signal as it is received at the receiver [114]. Ideally, in a direct Line of Sight<sup>1</sup> (LOS) scenario the power of the signal is approximately  $1/d^2$ , where  $d$  is the distance between the transmitter and the receiver. However, environmental and other factors can (and do) affect RSS measurements making it nonlinear. RSS is sometimes referred to as RSSI (Received Signal Strength Indicator/Indication) when used in cellular network context.

**Time of Arrival (TOA).** TOA, also known as Time of Flight (TOF), is a measure of the time a signal travels from an object (transmitter) to another (receiver). In order to correctly calculate TOA, both the transmitter and the receiver have to be synchronised appropriately, either by referring to a global clock or by exchanging time synchronisation information. Once the TOA is measured, the distance between the two objects can be estimated by the classical distance equation:  $d = v \times t$  where  $d$  is the distance,  $v$  is the speed of the signal and  $t$  is TOA. In free space, the speed of the signal is approximately equal to the speed of light ( $\approx 300,000 \text{ km/second}$ ).

The accuracy of both RSS and TOA largely depends on environment modelling. In urban environments, the Non-Line of Sight (NLOS) situation is very common where obstacles (natural or human-built) block the direct path between the transmitter and the receiver; albeit expensive, ways to mitigate the effect of NLOS exist [24]. Round Trip Time of Arrival (RT-TOA) [90] is a variant of TOA employed in systems where full time synchronisation is not possible or guaranteed. Basically, in RT-TOA an object  $P_1$  sends a signal to another object  $P_2$  at time  $t_1$ .  $P_2$  then replies back to object  $P_1$  which records the time it receives the reply, say  $t_2$ . Then, the RT-TOA is approximately  $t_2 - t_1$ . Usually, RT-TOA neglects some delays, such as processing delay, so RT-TOA can be used as long as these delays do not significantly affect the estimation accuracy, otherwise they should be modelled and accounted for during the calculation.

**Time Difference of Arrival (TDOA).** In TDOA, only reference objects have to be synchronised [50], this is in contrast to TOA measurements where synchronisation

<sup>1</sup>The transmission between two entities is said to be in a line of sight when the signals are not blocked or affected by obstacles on their way from the transmitter to the receiver.

is required among both the reference and the localised objects. TDOA measures the time difference of a signal received at different reference objects. To localise an object in 2 dimensions, at least 3 reference objects are required (4 reference objects for 3 dimensions). First, the TDOA between the target object and two reference objects form a hyperbola where the target object is located, with the two reference objects being foci<sup>2</sup>. A third reference object adds a second hyperbola where the intersection of the two hyperbolas is the location of the target object. TSOA (Time Sum of Arrival) is based on similar principles, but in this case the sum of the TOA at several reference points form ellipsoids intersecting at the target's location [95].

**Angle of Arrival (AOA).** In AOA, the reference objects measure the angle between the arriving signals emitted by the target object and a reference direction known as the *orientation* [109]. In order for the reference objects to be able to make AOA measurements, they need to be equipped with *antenna arrays*, which not all standard sensor nodes are usually equipped with; this makes adopting AOA as a primary location technique in ad hoc and Personal Area Network (PAN) slightly expensive. AOA is severely affected by the NLOS conditions; under these conditions, signals received are not necessarily coming from the direction where they were originally transmitted from by the target. Another way to measure AOA is when a receiving object has more than one integrated directional antenna, in which case the AOA is the RSS ratio between at least two of the antennas [103], but this is still a rather expensive solution.

### 2.2.2 Geometric Location Estimation

Once parameter measurements are obtained as detailed above, the location of the target object can be geometrically estimated by:

1. triangulation (based on AOA measurements),
2. trilateration (based on TOA or RSS measurements),
3. multi-lateration (based on TDOA measurements),
4. multi-angulation (based on multiple AOA measurements).

**Triangulation.** In triangulation [63], an object is localised based on AOA measurements from two reference objects. Figure 2.1 illustrates a standard triangulation procedure, where  $C$  is triangulated in reference to  $A$  and  $B$ . Since we assume knowledge

---

<sup>2</sup>In a hyperbola there are two foci points,  $F_1$  and  $F_2$ . These points have the property that given any point  $x_i$  on either of the hyperbola's curves, the difference between the distance from  $x_i$  to  $F_1$  and  $x_i$  to  $F_2$  is constant.

of the locations of  $A$  and  $B$ , the distance between them can easily be calculated by the following formula:

$$\overline{AB} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \quad (2.1)$$

where  $(x_a, y_a)$  and  $(x_b, y_b)$  are the coordinates of  $A$  and  $B$ , respectively. After measuring the angles  $\alpha$  and  $\beta$  (see to figure 2.1), and calculating the distance from  $A$  to  $B$ , we can use the law of sines to obtain the distance between  $A$  and  $C$ , denoted  $\overline{AC}$ , or the distance between  $B$  and  $C$ , denoted  $\overline{BC}$ , (either will suffice) as follows:

$$\frac{\sin \alpha}{\overline{BC}} = \frac{\sin \beta}{\overline{AC}} = \frac{\sin \delta}{\overline{AB}}$$

then solving for  $\overline{AC}$ , we have:

$$\overline{AC} = \frac{\overline{AB} \cdot \sin \beta}{\sin \delta} = \frac{\overline{AB} \cdot \sin \beta}{\sin(180 - \delta)} = \frac{\overline{AB} \cdot \sin \beta}{\sin(\alpha + \beta)}$$

since  $\alpha + \beta + \delta = 180$  (hence,  $180 - \delta = \alpha + \beta$ ) and  $\sin \delta = \sin(180 - \delta)$ , then  $\sin \delta = \sin(\alpha + \beta)$ . Now, we need to find  $\overline{XC}$ , such that the triangle  $\widehat{AXC}$  is a right triangle. Since we know  $\overline{AC}$ , then:

$$\overline{XC} = \overline{AC} \cdot \sin \alpha$$

knowing both  $\overline{AC}$  and  $\overline{XC}$ , we can now easily obtain  $\overline{AX}$  using Pythagoras theorem:

$$\overline{AX} = \sqrt{(\overline{AC})^2 - (\overline{XC})^2} = \sqrt{\left(\frac{\overline{AB} \cdot \sin \beta}{\sin(\alpha + \beta)}\right)^2 - \left(\frac{\overline{AB} \cdot \sin \beta \cdot \sin \alpha}{\sin(\alpha + \beta)}\right)^2}$$

Finally, the coordinates of  $C$  are determined in reference to the coordinates of  $A$  (or similarly  $B$ ), such that  $x_c = x_a + \overline{AX}$  and  $y_c = y_a + \overline{XC}$ .

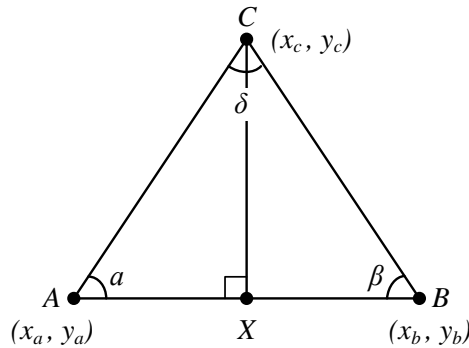


Figure 2.1: Triangulation

**Trilateration.** The simplest way to trilaterate a target is by solving a system of at least 3 quadratic equations consisting of the circle equations of three intersecting circles representing the reference objects. These circles are formed by measuring the distances between the reference objects and the target, which constitute the radii of the circles (each reference object forms one circle). Solving such system yields the intersection point of the circles and this is where the target is located (see section 4.3.1). However, in most cases, the three circles will not be ideally aligned to intersect in exactly one point. Instead, they will probably intersect in three points forming a *circular triangle* (also called curvilinear triangle) as shown in figure 2.2. Fewell [38] proposed an algorithm to calculate the common overlap area when three circles intersect. Since we are only interested in finding the three intersection points (not the actual area bounded by them), we use Fewell’s algorithm up to the stage where the intersection points are calculated, then we treat them as vertices of a regular triangle and find its centroid, where the target probably is. Figure 2.2 illustrates how trilateration is performed, where  $r_a$  is the radius of circle  $a$ ,  $d_{ab}$  is the distance between the centres of circles  $a$  and  $b$ , which can be calculated from equation 2.1 (since the locations of the reference points are known), and  $(x_{ab}, y_{ab})$  is the intersection point<sup>3</sup> between circles  $a$  and  $b$ . Based on [38] and figure 2.2, we trilaterate the target as follows (detailed derivation of the equations can be found in [38]):

1. First, we calculate the sines and cosines of angles  $\theta'$  and  $\theta''$ .

$$\begin{aligned} \cos \theta' &= \frac{d_{12}^2 + d_{13}^2 - d_{23}^2}{2d_{12}d_{13}}, & \sin \theta' &= \sqrt{1 - \cos^2 \theta'} \\ \cos \theta'' &= -\frac{d_{12}^2 + d_{23}^2 - d_{13}^2}{2d_{12}d_{23}}, & \sin \theta'' &= \sqrt{1 - \cos^2 \theta''} \end{aligned}$$

2. Next, we calculate the three intersection points  $(x_{12}, y_{12})$ ,  $(x_{13}, y_{13})$ ,  $(x_{23}, y_{23})$ .

- $(x_{12}, y_{12})$ :

$$x_{12} = \frac{r_1^2 - r_2^2 + d_{12}^2}{2d_{12}}, y_{12} = \frac{1}{2d_{12}} \sqrt{2d_{12}^2 (r_1^2 + r_2^2) - (r_1^2 - r_2^2)^2 - d_{12}^4}$$

- $(x_{13}, y_{13})$ :  $x_{13} = x_{13}' \cos \theta' - y_{13}' \sin \theta'$ ,  $y_{13} = x_{13}' \sin \theta' + y_{13}' \cos \theta'$

where,

$$x_{13}' = \frac{r_1^2 - r_3^2 + d_{13}^2}{2d_{13}}, y_{13}' = \frac{-1}{2d_{13}} \sqrt{2d_{13}^2 (r_1^2 + r_3^2) - (r_1^2 - r_3^2)^2 - d_{13}^4}$$

<sup>3</sup>Two circles intersect in two points, but we are only interested in the point contributing a vertex to the circular triangle formed by the intersection with a third circle.

- $(x_{23}, y_{23})$ :  $x_{23} = x_{23}'' \cos \theta'' - y_{23}'' \sin \theta'' + d_{12}$ ,  $y_{23} = x_{23}'' \sin \theta'' + y_{23}'' \cos \theta''$   
where,

$$x_{23}'' = \frac{r_2^2 - r_3^2 + d_{23}^2}{2d_{23}}, y_{23}'' = \frac{1}{2d_{23}} \sqrt{2d_{23}^2 (r_2^2 + r_3^2) - (r_2^2 - r_3^2)^2 - d_{23}^4}$$

3. Finally, once the three vertices of the circular triangle is calculated, we treat it as a regular triangle and calculate its centroid which will be our estimated location of the target.

$$C = \left( \frac{x_{12} + x_{13} + x_{23}}{3}, \frac{y_{12} + y_{13} + y_{23}}{3} \right) \quad (2.2)$$

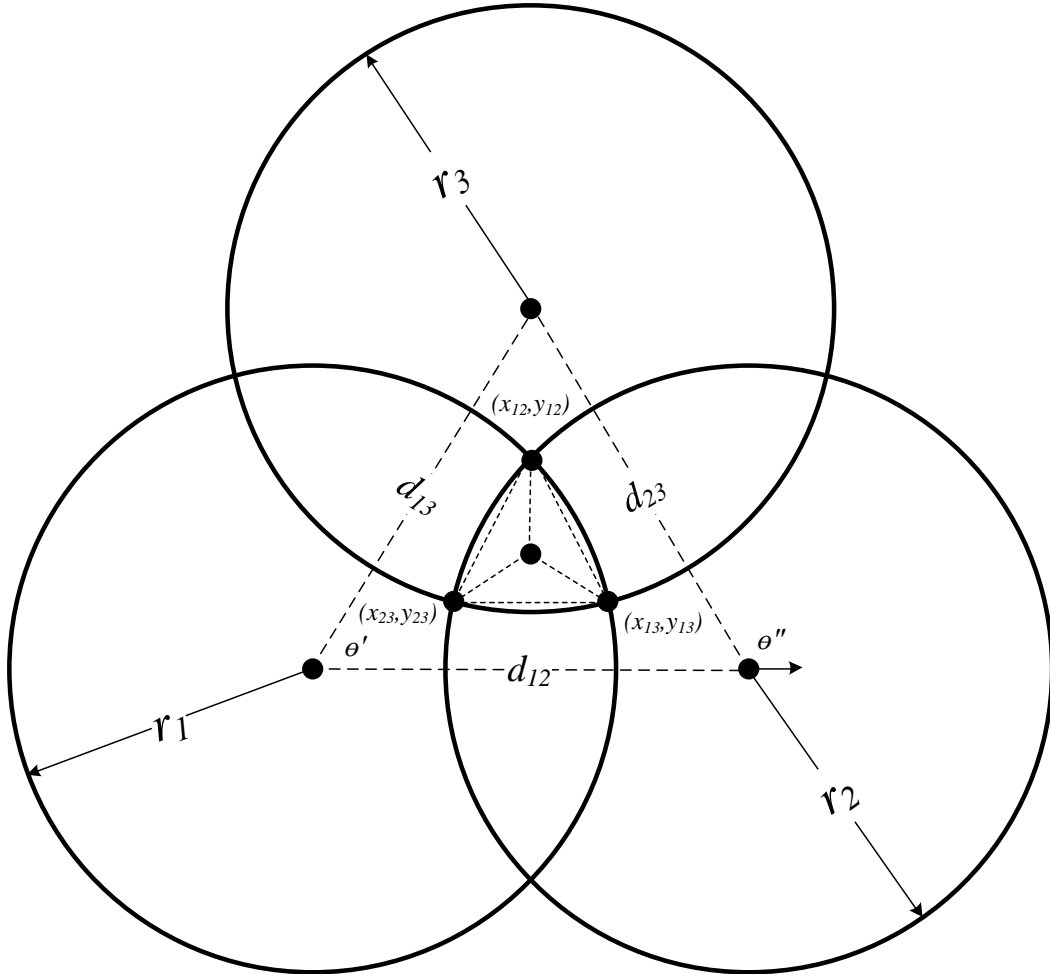


Figure 2.2: Advanced Trilateration

**Multi-lateration and Multi-angulation.** Multi-lateration [117] is similar to tri-lateration but is based on TDOA measurements rather than TOA or RSS. Generally, in multi-lateration, three reference objects measure the TDOA when receiving a signal from a target object which forms two hyperboloids intersecting at the target's location (TDOA from a fourth object forming a third hyperboloid is measured if localising the target object in 3 dimensions is required). Similarly, multi-angulation [12] is closely related to triangulation where a target object is localised based on known angles. However, while triangulation is localising a target object in reference to two objects, multi-angulation is a generalisation with more reference objects. Increasing the number of reference objects is especially beneficial to enhance the accuracy of the localisation process in noisy environments.

## 2.3 Localisation in Cellular Networks

In 1996, the Federal Communication Commission (FCC) issued the E911 mandate aiming to further improve the 911 emergency calls when being made from mobile handsets. The mandate requires telecom operators to be able to accurately locate a mobile handset initiating a 911 call with accuracy of around 50 meters 67% of the time and within 150 meters 95% of the time for handset-based (i.e., object-based) solutions, and within 100 meters 67% of the time and within 300 meters 95% of the time for network-based solutions [37]. Similarly, since July 2006, Ofcom (the telecommunication regulatory authority in the UK) gave support to caller localisation made via GSM/UMTS for emergency calls dialled to several emergency numbers (the European equivalent of 911). The service is limited to identifying the cell in which the call was made, which may cover a range of several meters in urban areas but can extend over several kilometres in rural ones. Below we discuss a few popular localisation techniques in cellular networks (some techniques such as Cell ID and Enhanced Cell ID are not discussed due to their severe accuracy discrepancies; see [17] for more information).

**Enhanced Observed Time Difference.** E-OTD [77] is based on Observed Time Difference (OTD) measurements, which estimate the time difference of receiving multiple signals. In E-OTD, the Mobile Station (MS) estimates its own location by calculating OTD when receiving signals from pairs of Base Stations (BS). This requires at least three reference BS' ( $P_1, P_2, P_3$ ) to be able to make at least two OTD measurements (e.g.,  $OTD_1$  from  $P_1$  and  $P_2$ ,  $OTD_2$  from  $P_1$  and  $P_3$ , or  $P_2$  and  $P_3$ ). While E-OTD is often used in GSM networks, OTDOA (Observed Time Difference of Arrival) is generally considered the UMTS version of E-OTD developed especially to operate on UMTS

networks. E-OTD was strongly believed to be the next generation of location service. However, beside requiring the handsets to be slightly modified to enable E-OTD (introducing cost implications), it also failed to meet FCC E-911 location performance requirements. Recently, E-OTD/OTDOA were largely replaced by U-TDOA.

**Uplink Time Difference of Arrival.** U-TDOA [98], standardised by 3GPP (3rd Generation Partnership Project), is a localisation technology that estimates the location of a MS by measuring how long it took signals emitted from the MS to be received at several BS'. Unlike E-TOA, U-TDOA is a network-based technique; that is, the localisation process is carried out at the reference BS' which does not impose extra hardware or software requirements on the MS'. Moreover, U-TDOA uses multi-lateartion for improved accuracy (see section 2.2.2).

**Global Positioning System.** GPS [107] is a location system developed by the US Department of Defence (DoD). GPS is similar to E-OTD in that it is handset-based (the target localises itself in reference of surrounding reference objects), but in this case the reference objects are satellite rather than BS'. To be able to use GPS, an object has to be equipped with a special GPS receiver to correctly receive and decode signals from at least 4 out of the 24 satellites orbiting the earth that are constantly emitting these GPS signals. GPS localisation and tracking proved to be useful for certain situations, but beside requiring additional hardware, it is also not suitable for indoor or underground environments where GPS signals are usually not available.

**Assisted Global Positioning System.** With conventional GPS systems, the GPS device localises itself independently, from receiving the GPS signals to the location estimation. An A-GPS [34] system, on the other hand, consists of three components:

1. an A-GPS devices that can receive GPS signals but can not decode them,
2. an A-GPS server equipped with a fully featured GPS receiver, and
3. a network infrastructure mediating between the A-GPS devices and A-GPS server.

Briefly, there are two types of A-GPS. In terminal-based A-GPS, the A-GPS device localises itself by sending the GPS signals it receives to the A-GPS server, which, based on these signals, carries out the localisation estimation and sends it back to the device. In network-based A-GPS, the A-GPS server provides some information (e.g., reference locations, reference time etc.) for the A-GPS device to carry out the location estimation. Recently, the enhanced A-GPS has been introduced, where cellular networks are used to improve the localisation efficiency.

**Differential Global Positioning System.** In most cases, GPS signals received from the satellites contain timing errors [97], which affect the accuracy of the location estimation. Usually, two receivers located within approximately the same vicinity receive the same timing errors. Thus, in D-GPS reference stations with pre-configured accurate locations are distributed carefully to cover a large area. Once these static stations receive GPS signals, they compare the signals with their pre-configured locations to find any timing errors. Information about these errors is then propagated to the mobile stations in their vicinity so they, in turn, can correct their received GPS signals.

## 2.4 Localisation Fusion

Multi-sensor data fusion entails combining data from different sources to improve the estimation accuracy of a particular process [51]. Accordingly, fusing more than one localisation technique/measurement proved very efficient [76]. The classical example is when a moving object is simultaneously detected by a pulsed radar and an infrared imaging sensor [85]. The radar can reliably measure the range of the object, but cannot accurately measure the angular direction. In contrast, the infrared sensor can accurately measure the angular direction, but not the range. Thus, by fusing measurements from the two sensors, we significantly improve the localisation of the object. Fusion, however, often introduces additional overhead, which is not desirable for low cost sensor networks. Below we discuss a few examples of fusion algorithms in wireless networks.

### 2.4.1 Fusing Different Technologies

Localisation accuracy can be improved by fusing the outputs of multiple wireless technologies where possible. For example, in [10] and [11], Aparicio *et al.* proposed an algorithm to locate a target in an indoor environment by fusing Bluetooth and WLAN measurements where Bluetooth stations and Access Points (AP's) are randomly distributed over an area. This technique incorporates building two maps, one based on RSS measurements from Bluetooth stations and another based on RSS measurements from WiFi AP's. The main idea is to specify the boundaries of the localisation area by Bluetooth (which is a short range technology and would produce more accurate estimates for this purpose) and then only accept the WiFi RSS measurements reporting the target to be within this area.

Another yet more advanced fusion system is COMPASS (COMMon Positioning Architecture for Several Sensors) [71] that fuses the outputs of multiple sensors to produce probabilistic location estimates by means of probability distribution function, PDF. COMPASS can fuse location information taken from multiple location sources,

including access points, GPS, RFID and Gyroscope. Basically, COMPASS is a software architecture based on modular plugin design, where each location source has a separate plugin. The plugins calculate location PDFs based on data provided by their corresponding location source and report these PDFs to a central *locator*. The locator then combines the received PDFs and calculates the location with highest probability. In [72] Kargl *et al.* provided an extension to the COMPASS system, with this extension raw geo-location information is translated to meaningful location information, such as city name, addresses etc., that can be readily used by location-based services.

### 2.4.2 Fusing Different Parameters

Fusing different technologies is certainly desirable, but sometimes it is not possible. Thus, the most common approach in multi-sensor data fusion is to fuse the measurements of different parameters, such as RSS and TOA. In the following subsections we briefly discuss such fusion techniques. Algorithms proposed to fuse multiple measurements of the same parameter at different intervals, e.g., [136], are not discussed.

**Fusing Signal Strength with Time Measurements.** Catovic *et al.* [22, 23] proposed an algorithm to fuse TOA/TDOA measurements with RSS in short-range partially synchronised Wireless Sensor Networks (WSN). The algorithm benefits from the short-range nature of WSN, which improves TOA/TDOA and RSS measurements. The proposed algorithm also accounts for the heterogeneous characteristics of WSNs, which influences some general communication properties such as communication range and routing. The main drawback of this algorithm is that it requires the nodes to be partially synchronised is not always possible in WSNs. Similarly, Luo *et al.* [88] proposed a self-localisation algorithm based on Covariance Intersection<sup>4</sup> (CI), which fuses RSS and TDOA measurements.

**Fusing Direction with Time Measurements.** In [127], Venkatraman *et al.* proposed two algorithms based on TOA and AOA fusion. The first algorithm, called Hybrid TOA/AOA Algorithm, is based on trilateration where a target object is located at the common overlap area of at least three intersecting circles formed by TOA measurements from at least three reference objects. In this algorithm, AOA measurements are taken to further constrain this area and enhance the accuracy of the localisation. The second algorithm, called Hybrid Lines of Position Algorithm, is based on solving

<sup>4</sup>CI fuses two or more variables with unknown correlation.

Lines of Position (LOP)<sup>5</sup> by the least square algorithm<sup>6</sup>. LOP are generated by an astronomical method called Intercept Method that is usually used to locate an object on earth. Similarly, Cong *et al.* [28] proposed a two-step least square algorithm to fuse TDOA and AOA measurements in wide-band CDMA cellular network. Additionally, Hsin-Yuan *et al.* [65] and Ping *et al.* [105] proposed other algorithms to fuse angular (AOA) and time (TOA/TDOA) measurements with neural networks.

## 2.5 Accuracy Issues

Maintaining consistent estimation accuracy is the main problem in most localisation techniques. The ideal situation of having a clear line of sight between the transmitter and the receiver rarely occurs in reality, especially in urban environments. In fact, localising an object in practice is often based on inherently nonlinear parameters that are easily affected by environmental and physical factors. In the case of RSS, for example, the strength of the received signals may not perfectly correlate with the distances it travelled because they may have been reflected, deflected or absorbed by en-route obstacles. However, careful investigating of the signalling used in a particular technology may lead to a better estimation; for example, recent work [64] showed that in Bluetooth measuring the received power level usually yields better location estimation than other Bluetooth parameter measurements. Another solution is to use filters, which basically estimate or recover missing or corrupted signal parameters. Nevertheless, using filters can be computationally expensive and may not be suitable for some low-end technologies, such as wireless sensor networks.

As discussed earlier, the localisation algorithms are as accurate as the parameters they are based on, and the accuracy of the latter mainly depends on correct modelling of the measured radio waves. Radio waves are usually described by their behaviour while propagating from a point to another. Modelling these radio propagation behaviours largely influences the accuracy of any localisation process. Based on the environment, radio propagation models can be roughly classified as follows [116]:

- Foliage Models: radio propagation through foliage,
- Terrain Models: effect of terrain characteristics on radio propagation, and
- City or built-up Models.

---

<sup>5</sup>A single LOP is a line in which a target object is situated. The intersection of multiple LOP yields the location of that target.

<sup>6</sup>The least square algorithm solves systems in which the number of equations is greater than the number of unknowns (such systems are called overdetermined systems).

City Models are derived from empirical data collected at urban environments, primarily to investigate radio propagation characteristics in such environments. Young Model, Okumura Model, Hata Model and Lee Model are a few examples of popular city radio propagation models [116]. These models, however, describe long-range propagation and hence can be used in localisation in long-range networks, such as cellular networks; for a discussion about radio propagation models in short-range environments, see [35].

**Integrity of Evidence.** It is clear that the vast majority of localisation/tracking mechanisms (especially the passive ones) bear probabilistic distribution due to the unavoidable measurement errors imposed by the environment. In one hand, it seems that pinpointing the exact location of a target entity with 100% accuracy is infeasible (if at all possible) in practice, on the other hand, if localisation is carried out as part of forensic investigation, it is important to characterise the estimation errors carefully as the produced evidence will probably be used in court. In this thesis, while we tried to design our algorithms to maintain acceptable accuracy, highly reliable measurements may not always be possible. Thus, we further discussed factors and methods that will most likely improve our accuracy, e.g., sections 5.7 and 6.5. However, in general it should be noted that the algorithms presented in this thesis, like most other tracking algorithms, produce probable evidence, not 100% factual, and that the amount of available resources will play a major role in determining whether or not they can be used in court. Nevertheless, probability distribution functions (PDFs) can be used to accurately characterise errors, which, ideally, should exhibit uniform distribution. Alternatively, this can be done by adopting measurement models that, e.g., can calculate the Cramér-Rao bound (CRB)<sup>7</sup> of the precision of the location estimation [103].

## 2.6 Summary

In this chapter, we surveyed various localisation techniques in wireless networks. Generally, localisation can be either object-based or network-based. In both cases, a target is localised in reference to several surrounding objects with known locations. In object-based localisation, the target localises itself by observing emissions from its neighbouring objects, while in network-based localisation, these neighbouring objects localise the target by observing its emissions. We first discussed several generic localisation techniques in sensor and cellular networks, we then provided a discussion about multi-sensor data fusion where various localisation parameters are fused to improved accuracy.

---

<sup>7</sup>Informally, the CRB is a lower bound describing how much information a set of measurements have about a parameter with unknown probability distribution.

## Chapter 3

# Forensic Tracking and Mobility Models

*Traditional tracking applications were mainly civilian such as tracking of goods and children. Military tracking applications are also popular and mostly based on Radar technology. A third type of tracking applications is conducted under a law enforcement investigatory framework to collect forensic evidence associated to some criminal activity, we call this kind of tracking “forensic tracking”. In this chapter, we first define and elaborate on this (relatively) new class of tracking and introduce some approaches that can be adopted to conduct forensic tracking procedures. We then describe several categories of mobility models and illustrate how they are developed in practice. Mobility models are especially important to conduct offline forensic tracking and will be used extensively in chapter 7. Parts of this chapter were published in [1] and [8].*

### 3.1 Introduction

Forensic tracking<sup>1</sup> is the tracking of individuals for forensic purposes. These individuals are usually suspected to be/have been associated with a particular crime (or set of crimes). Like other classes of tracking, forensic tracking can either be online (takes place in real time) or offline (takes place post hoc), but unlike other tracking classes, forensic tracking is always passive because it takes place clandestinely without the knowledge of the individual being tracked. While conventional civilian tracking applications usually

---

<sup>1</sup>Forensic tracking is also sometimes used to refer to the process of embedding marks in a digital content to track illegal redistributions [82].

use active online tracking in the sense that the target is aware of the tracking process and indeed sometimes cooperates with the tracker, in forensic tracking it is crucial that the tracking process is totally passive and unobservable by the target. Offline tracking, nevertheless, is naturally passive since the tracking process takes place after the fact where a collection of traces (corresponding to locations where the target was observed) are obtained and probabilistically reconstructed; obviously, in most cases, the whole trace collection process happens without the knowledge of the target.

**Chapter Outline.** This chapter is organised as follows. In sections 3.2 and 3.3 we discuss online and offline tracking, respectively. Online tracking can further be active or passive, which are discussed in sections 3.2.1 and 3.2.2, respectively. We then briefly describe several attacks against online tracking in section 3.2.3, followed by some privacy concerns associated with online tracking in section 3.2.4. Since offline tracking cannot be active, we only discuss passive offline tracking. In sections 3.3.1 and 3.3.2, we discuss offline tracking based on the popular Bayesian approach. Finally, in section 3.4 we provide an overview of mobility models describing their different types, and illustrating how they are being formally developed.

## 3.2 Online Tracking

Online tracking, also sometimes called live tracking, is the tracking of a target (or a number of targets) in real time. Online tracking has many civilian and military applications and can either be active or passive. In the following sections, we elaborate on both types and discuss example applications of each.

### 3.2.1 Active Tracking

The majority of tracking applications are based on active online tracking, and indeed, most of the literature in tracking covers the various aspects of active tracking. In this kind of tracking, the target is aware of the tracking process, and in some cases even collaborates with the tracker to localise itself. This means that active online tracking cannot be considered a type of forensic tracking. For completeness, however, below we, very briefly, discuss some popular active online tracking applications.

**Tracking in Sensor Networks.** Since, usually, there are limited resources available for sensor nodes, it is important that they maintain an efficient power-saving scheme. It is, therefore, not surprising that most of the tracking algorithms proposed for sensor networks try to minimise power consumption as much as possible, e.g., [131, 132].

One approach is to minimise the number of the active tracking sensors to only those located closer to the target. Kim *et al.* [74] proposed an algorithm that tracks a target through a set of steps. Once the surrounding objects detect the presence of the target, they collaboratively localise it and predict its next movement based on its velocity, assuming that the target does not perform sudden or rapid movements. The nodes then notify other nodes located at the area toward which the target is approaching. When tracking multiple targets, the power efficiency requirement becomes even more challenging. Jiang *et al.* [70] proposed an algorithm to maintain efficient power consumption threshold in a multi-target tracking scenario. The algorithm divides the tracking area into tracking sub-areas where nodes frequently switch between sleep and awake states based on a scheduling scheme. Similarly, distributed tracking became an established approach [118] where the tracking area is divided into sensor cliques, but the tracking scene for such algorithms need to be pre-configured.

Binary proximity sensors are also used to track both single [75] and multiple [121] targets. These sensors produce a 1-bit output to indicate the presence of the target(s) in their vicinity. However, while algorithms based on binary sensors perform well in short-range networks, they do not scale for the long-range ones. Research in the feasibility of tracking multiple targets by binary proximity sensors is undergoing; in particular, currently, the binary proximity sensors can, with high probability, detect the presence of targets, but cannot *accurately* count or identify them.

**Tracking in Cellular Networks.** Tracking a mobile handset in cellular networks (including GSM/UMTS and CDMA) has always been an active area of research. However, because such tracking is based on long-range communication, the accuracy of algorithms developed for this purpose is severely hindered. Beside the conventional localisation methods adopted from sensor networks (see section 2.2.1), filtering is usually also used to remove noise and further improve the location estimation accuracy. For example, in [93], Mihaylova *et al.* presented two sequential Monte Carlo based techniques, namely, a particle filter and a Rao-Blackwellised particle filter, which are based on RSSI measurements of signals emitted by the Mobile Station (MS). Similarly, Zaidi *et al.* [135] and Olama *et al.* [100] proposed several object tracking algorithms based on Kalman and particle filters. However, generally these filters impose an additional computational requirement and hence such algorithms are usually network-based; that is, there is usually a dedicated computation centre to which nodes report the measurements to carry out the location estimation computation. This setting is feasible in cellular networks as they are infrastructure based networks. Sensor network, on the other hand, are ad hoc networks which makes such solutions unsuitable.

**Radar Tracking.** Although we will consider multiple target tracking in chapter 4 based on wireless mobile ad hoc networks, the majority of research in this area has been conducted based on radar technology [68], mostly for applications to aviation systems, where the problem of accurately tracking multiple targets is even more challenging, especially when associating anonymised measurements to construct the tracking records of each target [128]. Multiple Target Tracking (MTT) algorithms in radar can roughly be based on: Nearest Neighbour (NN), Joint Probabilistic Data Association (JPDA) [83] or Multiple Hypotheses Tracking (MHT) [108]. Detailed discussion about radar-based tracking is beyond the scope of this thesis.

### 3.2.2 Passive tracking

Unlike active tracking, in passive tracking the targets must not be aware of the tracking process. Most (if not all) law enforcement tracking applications enforce this passivity requirement since the targets are usually suspects or criminals that are being tracked as part of some criminal investigation. Passive tracking is indeed more challenging than active tracking because in this case we not only need to track the target efficiently and accurately, but also be unobservable by the targets. Thus, probably a better name for this kind of tracking is “clandestine tracking”, but we will stick with “passive tracking” since this is more widely accepted.

The process of tracking by passive sensors (sensors that measure the radiation emitted or reflected by surrounding objects) is also called passive tracking, but here we make an explicit distinction between this passive tracking and our passive tracking. Additionally, our passive tracking is also required to be adaptive to cope with the movements of the targets. Unfortunately, the literature severely lacks contributions in passive/clandestine tracking. Therefore, in this thesis, we tried to bridge this gap by extensively discussing and proposing passive tracking algorithms for different environments, which can be readily used by law enforcements.

### 3.2.3 Reliability and Security

While it is of course important for the tracking process in the conventional tracking applications to be reliable, this is even more emphasised in forensic tracking because the produced evidence will be used in court and thus its integrity needs to be tightly preserved (see section 2.5). Beside carefully characterising measurement errors, it is also important to be aware of how potential malicious attackers can masquerade the tracking process and hence the resulted forensic evidence. In the following subsections we discuss a few possible ways a tracking process can be maliciously attacked. We

describe attacks in the typical tracking scenario where a target is being tracked by a single or multiple trackers, but these also apply for multiple targets scenarios. We will later revisit some of these attacks in chapter 4 and propose lightweight cryptographic solutions; see section 4.7.

**Address Spoofing.** If the address of any of the genuine trackers was spoofed, the integrity of the whole tracking process fails. In such scenarios, an attacker can impersonate one of the trackers and overtake the tracking process. During this time, the attacker can easily modify the tracking information. This attack, however, can be prevented by enforcing mutual authentication between the trackers to prove to each other that they are in fact who they claim they are (see section 4.7.1).

**Denial of Service (DoS).** Another way to attack a tracking process is to temporarily disable it by temporarily rendering its resources unavailable. Such attacks involve sending frequent random traffic to the trackers, which may result in losing track of the target. This attack may be prevented by configuring the trackers to accept traffic from only specific entities (e.g., the targets and other trackers).

**Man-In-The-Middle (MITM).** An attacker can mediate between two or more trackers pretending to be one for the other. An attacker in this case can either be passive, where it only relays traffic, or active, where it alters traffic as it passes from one side to the other. This type of attack can be prevented by encryption and the various integrity check methods. Usually, the traffic is location updates and is small enough to allow encryption without necessarily overwhelming the tracking process.

### 3.2.4 Privacy Implications

In active tracking, the tracking process takes place either with the collaboration of the target, or at least with a prior consent. In both cases, there are no privacy implications as the target is already aware of and has authorised the tracking process. On the other hand, this is not the case with passive tracking because the targets are not aware that they are being clandestinely tracked. Additionally, as we will see in chapters 4 and 5, passive tracking may require the active propagation and distribution of software and the passive recruitment of tracking agents to assist in the tracking process, which may involve both ethical and legal issues. However, while in some jurisdictions, this kind of passive tracking may not be permitted, in others it may be. In the United Kingdom, for example, the Regulation of Investigatory Power Act 2000 (RIPA) discusses similar issues under a *lawful interception* framework, and, in some situations, allows

law enforcement officers to bypass privacy barriers having that appropriate procedures are followed. In other words, while in some countries violating the privacy of users is strictly illegal, in others this may be restrictively allowed for a specific period of time.

### 3.3 Offline Forensic Tracking

Online tracking may not always be possible usually due to resource constraints. In fact, in most real criminal cases, online tracking is not even an option. In these cases, an offline forensic tracking investigation is often initiated amid a crime that had already occurred. Offline tracking begins by obtaining an incomplete set of traces associated to a target (or a number of targets) then proceeds to reconstruct them probabilistically; thus, a more intuitive name for offline forensic tracking (which we will use extensively and interchangeably in the rest of the thesis) is *trace reconstruction*. Clearly, active offline tracking does not make sense; offline tracking is naturally post-hoc passive. There are a number of approaches that can be used to reconstruct a target’s trace, prominently, Bayesian approach as it offers a coherent probabilistic framework that suits our reconstruction task. However, trace reconstruction based on Bayesian approach may become infeasible for sufficiently large scenarios, in which case we can use variants of mobility models instead. We provide a brief discussion about both approaches below; Bayesian offline forensic tracking is the subject of chapter 6 while offline forensic tracking based on mobility models is the subject of chapter 7.

#### 3.3.1 Basic Bayesian Approach

Bayesian analysis is based on the popular Bayesian formula which basically calculates the probability that an event  $A$  will occur given that some other event  $B$  has already occurred, denoted  $\Pr(A|B)$ , and is usually called the posterior probability, formally:

$$\Pr(A|B) = \frac{\Pr(AB)}{\Pr(B)} = \frac{\Pr(B|A) \cdot \Pr(A)}{\Pr(B)}$$

where  $\Pr(AB)$  is the joint probability of  $A$  and  $B$  (the probability that both  $A$  and  $B$  will occur), the probability  $\Pr(A)$  is the prior probability and is subjectively derived to model any prior knowledge of event  $A$ , the probability  $\Pr(B|A)$  is the conditional probability of  $B$  given  $A$ , also called the likelihood, and the probability  $\Pr(B)$  is the marginal probability, which is a normalising factor to make sure that probabilities will sum up to 1 (adhering to the fundamental axioms of probability theory). Although Bayesian approach has been successfully used in online localisation and tracking [41], using it in an offline setting seems even more natural since in this setting we are

confronted with the missing data problem for which Bayesian approach can provide appropriate probabilistic treatment. When extracting a trace for an offline tracking analysis, the missing data that the trace will most likely exhibit form *gaps*. Bayesian analysis can help in connecting these gaps by assigning probabilities to the different possible routes through the gaps, then select the route with the highest probability as the one that the target most likely has taken through the gaps. In Bayesian formulation, this scenario can be stated as follows:

What is the probability that a particular route has been taken by the target, given that the target spent some known time while traversing the gap?

The time the target spent traversing a particular gap can easily be found by observing the end points of the gap which indicate what time the target was last observed before entering the gap, and then what time the target was next observed as it was leaving the gap. This can be formulated as follows:

$$\begin{aligned} \Pr(\text{Route}_i|\text{GapTime}) &= \frac{\Pr(\text{GapTime}|\text{Route}_i) \cdot \Pr(\text{Route}_i)}{\Pr(\text{GapTime})} \\ &= \frac{\Pr(\text{GapTime}|\text{Route}_i) \cdot \Pr(\text{Route}_i)}{\sum_i \Pr(\text{GapTime}|\text{Route}_i) \cdot \Pr(\text{Route}_i)} \end{aligned}$$

### 3.3.2 Dynamic Bayesian Networks

Bayesian Networks (BN) are a convenient way to graphically represent causal relationships among several variables, where some variables directly or indirectly influence each other. The main feature of BNs is their conditional independence, which states that a variable is conditionally independent from its non-descendent given its parents. For example, in figure 3.1,  $B_t$  is conditionally independent from  $C_t$  given  $A_t$ , denoted  $B_t \perp\!\!\!\perp C_t | A_t$ . If two variables are conditionally independent given their parents, then change in one does not affect the other if the value of the parent is known. Thus, the probability density function of a Bayesian Network  $X$  containing  $n$  variables  $X = \{x_1, x_2, \dots, x_n\}$  is calculated as follows:

$$\Pr(x_1, x_2, \dots, x_n) = \prod_{i \in 1}^n \Pr(x_i | pa(x_i))$$

where  $pa(x_i)$  is the set of the parent nodes of variable  $x_i$ . This is much more efficient than applying the conventional chain rule:

$$\Pr(x_1, x_2, \dots, x_n) = \Pr(x_1 | x_2, \dots, x_n) \cdot \Pr(x_2 | x_3, \dots, x_n) \cdots \Pr(x_{n-1} | x_n) \cdot \Pr(x_n)$$

Once we specify the gaps in a target trace, we run the BN to select the routes that the target most likely has taken through these gaps. As we traverse the various routes, the BN constantly updates its state to reflect on the probabilities of the traversed routes, this is called *belief propagation*. However, conventional static BNs are not suitable in this scenario since the network has to be updated according to dynamic variables that change over time. This means that not only belief propagation is required, but also a dynamic state update, where future states are affected by the current ones (i.e., essentially forming a Markov Chain). In such situations, Dynamic Bayesian Networks (DBN) [46] can be used instead. Figure 3.1 depicts a sample DBN with four variables, showing how they are updated from time  $t$  to  $t + 1$ . Usually, every variable at time  $t$  updates its copy at  $t + 1$  (e.g.,  $B_t \rightarrow B_{t+1}; D_t \rightarrow D_{t+1}$ ), but it is also possible for a variable at  $t$  to update other variables at  $t+1$  (e.g.,  $A_t \rightarrow A_{t+1}, C_{t+1}; C_t \rightarrow C_{t+1}, D_{t+1}$ ).

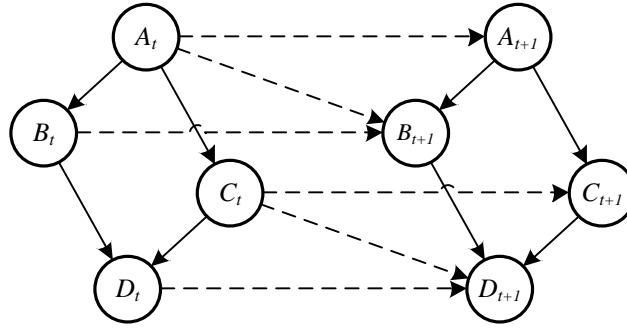


Figure 3.1: Sample Dynamic Bayesian Network (DBN)

These variables can represent factors affecting whether the target could have been observed at the location where the DBN is called. However, some of these factors may be fuzzy in nature. For example, one essential factor that a DBN should consider when calculating the probabilities is the road traffic states, but this factor is fuzzy since there is no clear distinction between what constitutes a low, medium or high traffic. Bayesian Networks utilising fuzzy variables are called Fuzzy Bayesian Networks (FBN) [44]. Combining FBN with DBN, we have the Fuzzy Dynamic Bayesian Network (FDBN), which seems to be a comprehensive framework for a typical Bayesian-based trace reconstruction.

### 3.4 Mobility Models

In sufficiently large scenarios (such as the one discussed in chapter 7), BN may not be feasible, in which case we can reconstruct the trace based on mobility models. Mobility models are mathematical descriptions of the motion behaviour of entities under some

constrains. These models are most commonly used in simulation-based evaluations of mobile and wireless network protocols. In our trace reconstruction setting, mobility models can be used to assist in estimating the target’s movement through the gaps. In particular, when we run a mobility model on a particular route, we are actually estimating the movement of the target should they have taken that route. That is, we supply the models with estimated information about factors that could have affected the target’s mobility and run it over the possible routes. In the following sections we briefly introduce different classes of mobility models and show how models in these classes are being developed to adapt for objects (nodes) with different mobility behaviours. The discussion in this section considers the standard mobility models (which we feel will provide essential background for the rest of the relevant parts of the thesis), and these may be used for trace reconstruction, but simplified variants of them (which we call delay mobility models) seem to do the job more efficiently and economically. Detailed discussion about trace reconstruction based on (delay) mobility models is deferred to chapter 7 where we use them in a multi-modal environment.

### 3.4.1 Pedestrian Mobility

Pedestrian mobility models [20] describe the motion behaviour of humans (pedestrians). Generally, these models are either discrete-space or continuous-space [80]:

- *Discrete-space*: in this approach the movements of the pedestrians are simulated in discrete time steps while representing simulation environment by a grid. The nodes (pedestrians) in these models move in a cellular automata fashion by rapidly switching between adjacent cells, thus they are also sometimes called cellular automata based models [115].
- *Continuous-space*: in these models, the nodes’ locations are updated continuously producing a smooth movement behaviour. Most of the models in this category model the pedestrian mobility based on the dynamics of fluids or gas [56].

One of the most popular pedestrian mobility models is the random waypoint, which was first proposed by Johnson and Maltz [30]. In this model each node  $n_i$  chooses a random destination  $d_{i,t}$  and approaches it in a random speed  $v_{i,t}$ . Once  $d_{i,t}$  is reached,  $n_i$  pauses for a random time  $p_{i,t}$ , chooses another random destination  $d_{i,t+1}$  and moves toward it with a random speed  $v_{i,t+1}$ , and so on until the simulation expires. This (somewhat basic) model has been extensively used in evaluating many ad hoc network protocols, for which mobility models are essential [133]. More realistic is the continuous-space approach which models pedestrians as Newtonian particles whose mobility is

affected by a number of *forces* [57, 58]; see figure 3.2 for an illustration. Helbing *et al.* modelled such forces in their famous social force model [59]. These forces are [87]:

- desire force  $\mathbf{F}^{\text{desire}}$ : models the node's self desire (motive) to reach its destination. Let  $n_i$  be a pedestrian with mass  $m_i$  intending to move with a velocity  $\hat{v}_i$  in the direction  $\vec{e}_i$ , then  $\mathbf{F}^{\text{desire}}$  is:

$$\mathbf{F}_i^{\text{desire}} = \frac{\hat{v}_i \vec{e}_i - v_i}{t_i} m_i \quad (3.1)$$

where  $v_i$  is  $n_i$ 's current velocity which he tries to adjust to reach  $\hat{v}_i$  within time  $t_i$ . Notice that equation 3.1 is derived from the popular force formula:  $F = m_i \times a$ , where the acceleration  $a = \hat{v}_i - v_i/t_i$ .

- social force  $\mathbf{F}^{\text{social}}$ : models the effect of other nodes  $\cup_{j \in \mathcal{N} \setminus \{i\}} n_j$  and the environment (e.g., physical obstacles)  $\cup_{i \in \mathcal{M}} m_i$  on the movement of node  $n_i$ , where  $\mathcal{N}$  and  $\mathcal{M}$  are the sets of all nodes and all obstacles, respectively, in the simulation scene. Here, each object (pedestrian or physical obstacle) is modelled as a circle with centre  $c_i$  and radius  $r_i$ .  $\mathbf{F}^{\text{social}}$  is calculated as follows:

$$\begin{aligned} \mathbf{F}_i^{\text{social}} &= \sum_{j \in \mathcal{N}, j \neq i} A_j \exp \left[ \frac{(r_i + r_j) - d_{i,j}}{B_j} \right] \vec{e}_{i,j} \\ &+ \sum_{k \in \mathcal{M}} A_k \exp \left[ \frac{(r_i + r_k) - d_{i,k}}{B_k} \right] \vec{e}_{i,k} \end{aligned}$$

where  $A_y$  and  $B_y$  are the magnitude and range of the force,  $y \in \mathcal{N} \cup \mathcal{M}$ ,  $d_{i,j}$  is the distance between object  $u_i$  and  $u_j$ , and  $\vec{e}_{i,j}$  is the unit vector pointing from  $u_i$  to  $u_j$  (the direction of the movement).

- physical force  $\mathbf{F}^{\text{physical}}$ : models the constrained mobility of nodes that is influenced by the physical structure of pedestrian pathways; this force also accounts for the avoidance behaviour among nodes.

$$\begin{aligned} \mathbf{F}_i^{\text{physical}} &= \sum_{j \in \mathcal{N}, j \neq i} A_j \cdot g((r_i + r_j) - d_{i,j}) \vec{e}_{i,j} \\ &+ \sum_{k \in \mathcal{M}} A_k \cdot g((r_i + r_k) - d_{i,k}) \vec{e}_{i,k} \end{aligned}$$

where  $A_y$  is the magnitude of the force,  $y \in \mathcal{N} \cup \mathcal{M}$ , and the function  $g(\cdot)$  is defined as follows:

$$g(x) = \begin{cases} 1 & \text{if } (r_i + r_j) - d_{i,j} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

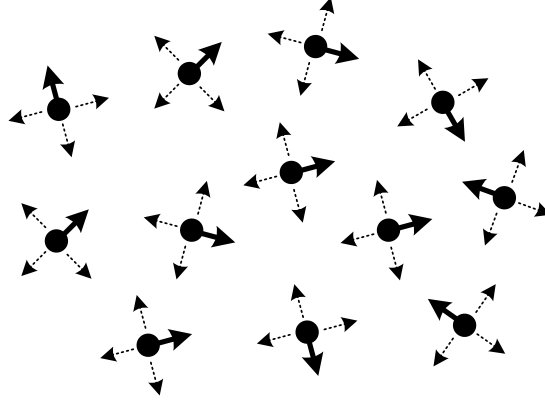


Figure 3.2: Forces determining pedestrian mobility

All these forces affect the pace at which the pedestrians move toward their destinations, which can be calculated by summing all the forces (recall,  $F = m \cdot a$  and  $a = \frac{d}{dt}v$ ).

$$\frac{d}{dt}v_i = \frac{\mathbf{F}_i^{\text{desire}} + \mathbf{F}_i^{\text{social}} + \mathbf{F}_i^{\text{physical}}}{m_i}$$

### 3.4.2 Vehicular Mobility

Modelling the mobility of vehicles<sup>2</sup> has traditionally been based on pedestrian mobility models [113]. However, due to the substantial intrinsic physical differences between pedestrians and vehicles, pedestrian mobility models often do not accurately capture realistic vehicular mobility patterns. Beside the fact that vehicles are significantly faster than pedestrians, they also have less motion freedom since they are usually constrained by roadway structures. This has motivated the development of dedicated vehicular mobility models, which can be roughly classified in four classes:

- *Macroscopic models*: simulate the characteristics of the roadways (motion constraints) and the flow of vehicles. These models do not consider the individual movement of the vehicles, but rather model the traffic in an abstract level.
- *Microscopic models*: describe the dynamics of the movement of individual vehicle entities and their interaction with each other. This is the most common class.
- *Mesoscopic models*: a hybrid class between macroscopic and microscopic models mixing features from both with moderate complexity, upper bounded by the complexity of microscopic models.

<sup>2</sup>Vehicles in this context includes both private and road-based public transport vehicles because they are all susceptible to the same mobility constraints on the road.

- *Sub-microscopic models*: also called nanomodels, these are very elaborate models simulating parts of the vehicles, such as the engine, the transmission systems etc. These models are the most computationally demanding.

Regardless of the class, the most desirable mobility modelling approach is to statistically derive a model from real-life traces [67]. Unfortunately, it usually takes long time and substantial effort to collect such traces. Thus, most vehicular mobility models are based on mathematical formulation, such models are called *synthetic models* and are classified as follows [40]:

- *Stochastic models*: these are microscopic models simulating vehicular movement with random speed over street map graphs; these models are the most trivial because they do not represent realistic vehicular behaviours such as interactions between vehicles and intersection overhead. Examples of such models are the City Model [31], and Manhattan Model [13].
- *Traffic stream models*: these are microscopic models simulating the fundamental traffic parameters such as velocity, density and flow. The model proposed by Lighthill and Whitham [84] is one of the earliest (and most fundamental) examples of traffic stream models.
- *Car-following models*: these are microscopic models where the mobility of each vehicle is modelled in relation to the vehicle ahead [21]. In these models, the movement pattern of a particular vehicle is modelled relative to its distance from the leading vehicle as well as the absolute and relative speed of both vehicles.
- *Car/flow interaction models*: these models extend microscopic scenarios by introducing multiple traffic flows and considering the interaction between vehicles from different flows (this includes lane changing and intersection management).

The car-following approach of modelling the mobility of vehicles is probably the most established (and studied) approach dating back to the 1950s. One of the earliest car-following models is the GHR model [45], which is based on differential formulation. However, differential formulation-based car-following models are not scalable and would require excessive computational power for large scenarios. A more convenient car-following approach is based on time discrete steps [79]. An example of a simple time-discrete car-following model is as follows: let  $d_{safe}$  be the minimum safety distance between any two vehicles at any given time. Based on Figure 3.3, and considering vehicle  $S_i$ , this translates to the following rule:

$$\Delta x_i - L_{i+1} \geq d_{safe} \quad (3.2)$$

Other vehicles should preserve similar inequality. This safety distance is inspected regularly at every time step  $\Delta t$ . If this distance is abused, a *deceleration* procedure is called to decelerate the following vehicle at the next time step:

$$v_i(t + \Delta t) = v_i(t) - (v_{i+1}(t) - v_i(t)) \quad (3.3)$$

Otherwise, the vehicle may either choose to keep its current speed, or accelerate:

$$v_i(t + \Delta t) = \min[v_i(t) + (v_{i+1}(t) - v_i(t)), v_{max}] \quad (3.4)$$

where  $v_{max}$  is the maximum legally allowable speed and is road-dependant. A more sophisticated model would also consider lane changing and take-over behaviours of the vehicles. Such models require more conditions to be satisfied before undertaking the lane-changing/take-over actions. In all cases,  $v_{j-1}(t) < v_i(t)$  has to be satisfied to avoid collisions while changing the lane or overtaking the leading vehicle. The following two inequalities need to further be satisfied when changing the lane:

$$(x_{j-1}(t) - x_i(t)) - L_{j-1} \geq C_{safe} \quad (3.5)$$

$$(x_i(t) - x_{j+1}(t)) - L_{j+1} \geq C_{safe} \quad (3.6)$$

where  $C_{safe}$  is a sufficiently large gap between two vehicles to allow a third one to slip between them. On the other hand, while overtaking, the node has to do extra calculations to make sure that it can shift back to its original lane later. Assume that the acceleration  $a_i$  of vehicle  $S_i$  can be roughly estimated upon which the displacement (distance)  $d_i(\tau)$  after time  $\tau$  can be calculated to check the possibility of whether overtaking may take place during that time. Considering figure 3.3, vehicle  $S_i$  first predicts the distance between vehicles  $S_{i+1}$  (the leading of the current lane) and  $S_{j+1}$  (the leading of the adjacent lane) after time  $\tau$ ,  $d_{gap} = d(S_{j+1}) - d(S_{i+1})$ , where<sup>3</sup>:

$$d(S_{j+1}) = \frac{v_{j+1}(t) + a_{j+1} \cdot \tau}{\tau}$$

and similarly for  $d(S_{i+1})$ . Once  $d_{gap}$  is calculated, and in addition to the inequalities 3.5 and 3.6, the following inequalities are also tested:

$$(d_{gap} \geq C_{safe}) \wedge (a_{j-1}(t) < a_i(t)) \quad (3.7)$$

An overtake proceeds only if all conditions are satisfied.

<sup>3</sup>Since  $v = v_0 + a \cdot t$ , and  $d = v/t$ , then  $d = (v_0 + a \cdot t)/t$ , where  $d$  is the distance,  $v$  is the average velocity,  $v_0$  is the initial velocity,  $a$  is the acceleration and  $t$  is the time.

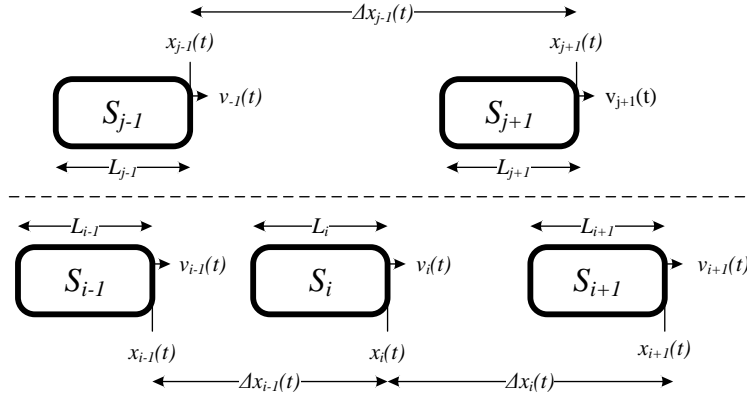


Figure 3.3: Scenario illustrating our sample vehicular mobility model

### 3.4.3 Multi-modal Mobility

Multi-modal mobility models describe the movement of nodes with irregular mobility behaviours over heterogeneous environments (environments where there is more than one way to transport from one point to another). In this case, we not only need to integrate several different mobility models, but also model the interaction between them, which turns out to be non-trivial. In particular, we need to model the Inter-Transport Delay (ITD) which is the transition delay incurred by switching between different mobility models. For example, considering a bus transportation network, for an individual to switch from bus  $a$  to bus  $b$ , the ITD is determined by the timetables of both  $a$  and  $b$ . We build the models based on such parameters, which are indeed publicly available. We represent these timetables by matrices listing all the locations that  $a$  and  $b$  pass by and all possible times; below is an example, where  $z_{i,j}$  is set to 1 if  $z$  appears in location  $j$  at time  $i$ , 0 otherwise, and  $z \in \{a, b\}$ .

$$\begin{bmatrix}
 a_{i,j} & a_{i,j+1} & a_{i,j+2} & \cdots & a_{i,y-2} & a_{i,y-1} & a_{i,y} \\
 a_{i+1,j} & a_{i+1,j+1} & a_{i+1,j+2} & \cdots & a_{i+1,y-2} & a_{i+1,y-1} & a_{i+1,y} \\
 \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
 a_{x,j} & a_{x,j+1} & a_{x,j+2} & \cdots & a_{x,y-2} & a_{x,y-1} & a_{x,y}
 \end{bmatrix}$$

$$\begin{bmatrix}
 b_{k,l} & b_{k,l+1} & b_{k,l+2} & \cdots & b_{k,q-2} & b_{k,q-1} & b_{k,q} \\
 b_{k+1,l} & b_{k+1,l+1} & b_{k+1,l+2} & \cdots & b_{k+1,q-2} & b_{k+1,q-1} & b_{k+1,q} \\
 \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
 b_{p,l} & b_{p,l+1} & b_{p,l+2} & \cdots & b_{p,q-2} & b_{p,q-1} & b_{p,q}
 \end{bmatrix}$$

These matrices are then used to calculate the ITD. Suppose that an entity arrived at location  $m$  at time  $t$  and needs to transit from  $a$  to  $b$ . This transition can be modelled using the following transition function (which ):

$$ITD_t^{a,b} = \begin{cases} |t - n| & \text{if } b_{n-1,m} < t < b_{n,m}, \\ 1 & \text{if } t = n, \\ \infty & \text{otherwise.} \end{cases}$$

This function basically models how long an entity needs to wait at location  $m$  before a carrier from  $b$  is available. Clearly, there are three cases, either the entity arrives at  $m$  between times  $n$  and  $n + 1$  (where  $n < t < n + 1$ ), or at exactly time  $n$ , or there is no  $b$  carrier at location  $m$  (e.g.,  $b$  terminates before  $t$  or does not pass by  $m$ ). The purpose of this section is not to propose a fully functional multi-modal model, rather it just aims to provide a proof of concept and a brief overview of how multi-modal models are developed. The above model is a simplification of a more detailed model proposed in chapter 7; also it may not cover all possible scenarios. Here we explicitly assumed that the transition location  $m$  will be present on the matrices of both  $a$  and  $b$ , which, obviously, does not necessarily always hold in practice; this requires a more elaborate case distinction that is beyond the scope of this introductory section. We also note that unlike the full model in chapter 7, here we only consider variants of vehicular models but without integrating a pedestrian one. We will revisit and elaborate on these and other issues in chapter 7 where we propose a full-fledge multimodal mobility model.

### 3.5 Summary

In this chapter, we discussed modern applications of tracking with special emphases on forensic tracking, where the latter entails tracking for forensic purposes. In general, tracking can either be online or offline. In online tracking, a target is being tracked in real time, this can further be active, with the knowledge of the target, or passive, without the knowledge of the target. While most online civilian applications are based on active tracking, online forensic tracking is always passive for obvious reasons. Offline tracking, on the other hand, entails the post hoc reconstruction of the target's trace; offline forensic tracking is more practically relevant in most criminal investigations. We discussed and described ways in which offline forensic tracking can be carried out, namely, using Bayesian approach or based on mobility models. In this chapter we briefly illustrated how to build and use these approaches to carry out an offline forensic tracking process, more details of each approach are deferred to chapters 6 and 7.

## Part II

# Online Forensic Tracking

## Chapter 4

# Online Pedestrian Forensic Tracking

*Target tracking applications have drastically evolved and became essential intelligence and law enforcement tools. Traditionally, long-range communications, such as cellular networks, have been used for this purpose. This conventional tracking approach, however, usually suffers from estimation inaccuracy due to measurement errors. On the other hand, short-range emissions from mobile devices such as phones and accessories can be used to improve the accuracy of these passive tracking applications. In this chapter, we adopt an algorithmic approach to build a clandestine agent-based passive tracking system where a set of dynamically recruited tracking agents observe single or multiple targets and report to single or multiple trackers. We develop a number of algorithms to initiate and maintain this tracking system while consistently preserving passivity and unobservability. We also propose a few supporting auxiliary mechanisms and algorithms to improve security and fault-tolerance. The contents of this chapter was published in [2], [7], and [3].*

### 4.1 Introduction

Modern tracking and surveillance applications are now being frequently utilised by law enforcements. Unlike the conventional tracking applications, law enforcement tracking applications are required to be passive since the targets in this case are suspects or criminals and it is important that they are not aware of the tracking process. While the ubiquitous use of mobile phones has contributed an important new localisation/tracking

tool, cellular-based tracking offers a limited temporal and spatial resolution causing severe accuracy implications in rapidly evolving scenarios where adding or switching between targets is desirable. This is particularly problematic inside buildings or at crowded urban environments where physical or cellular surveillance is often difficult (or impossible) without considerable resources and effort. On the other hand, short-range radio frequency emissions are typically used more often allowing for slightly easier (passive) tracking that does not require fixed infrastructure or interaction with network operators and can hence be set up rapidly on an ad hoc basis.

In this chapter, we approach passive tracking based on a secondary radio-frequency emanation associated with a number of personal electronic devices ranging from mobile phones to portable computers and their accessories, which use different radio frequency communication channels, including IEEE 802.11 and Bluetooth. The ubiquity of such devices and their use by the general public makes them attractive tracking tools. In fact, recent work [101] showed that the presence of such devices (Bluetooth devices in that case) is increasingly becoming common among the general public; they carried out their experiments in a relatively small-sized British city, so we would expect even higher number of such devices in larger cities. Even though we may occasionally borrow a few terminologies from Bluetooth technology (for convenience), we deliberately abstracted the algorithms making them generic enough to be applicable for any short-range ad hoc communication technology. Once an emission source associated to the target has been identified, typically in the form of a Layer 2 (MAC) address, the target can be localised by fusing observation reports from multiple agents and then geometrically transform these to a target location estimate.

This chapter contributes a number of novel algorithms for pedestrian tracking in a different settings and scenarios. We first propose algorithms for locating and tracking a single target by randomly and dynamically recruiting agents while forming a number of tracking networks interconnected in an ad hoc fashion. These agents are typically mobile phones and similar devices from the general public but may also include purpose-built units. These algorithms constitute a complete (basic) passive tracking system. We begin by proposing the core tracking functions and discussing a (well known) localisation technique that is being used by the tracking algorithms as a block-box every time an update of the target's location is required. We then proceed to extend our tracking scenario and add more functionalities to support multiple-target tracking. This potentially allows for detecting and tracking associates of targets or multiple devices carried by the same target. While such an individual may frequently change emission devices to obscure tracking, the mere fact that the same individual possesses other devices, such as other mobile phones, for a period of time may allow for

the formation of useful hypotheses. We also report on a group of algorithms to track multiple targets by (possibly) multiple trackers while minimising the observability of coordinating communication and maximising coverage and tracking accuracy among the dynamically recruited agents. We do not impose specific computational requirements on the tracking agents, while also not assuming that such agents are capable of carrying out heavy computations such as those requiring signal filtering. We instead tackle the problem of multiple targets tracking at a generic algorithmic high level.

**Chapter Outline.** The remainder of this chapter is organised as follows. Section 4.2 sets the scene for the rest of the chapter and discusses our assumptions and requirements; it also provides further background information that will be used in later sections. Online pedestrian localisation is discussed in section 4.3. Section 4.5 describe how tracking networks are formed, which is a crucial part of our tracking system since we adopt an agent-based tracking approach. We then propose our tracking algorithms in sections 4.6 and 4.7 considering scenarios ranging from single tracker single target (simulation results in section 4.6.2), to multiple trackers multiple targets scenarios (simulation results in section 4.7.6). In section 4.7.3 we describe how our tracking network handles failures, followed by an elaboration of the dynamic leader election algorithm in section 4.7.4, which is used throughout the tracking. To minimise observability, we also propose a transmission algorithm in section 4.7.5. Finally, privacy in online forensic tracking and ways to preserve it are discussed in section 4.8.

## 4.2 Scene Setting and Assumptions

In our proposed passive clandestine tracking system there are three types of entities:

- *Trackers*: entities initiating the tracking process. We assume that trackers are law enforcement officers or members of the police.
- *Targets*: entities that are being tracked by the trackers. We assume that targets are suspects or criminals being investigated as part of a crime investigation.
- *Agents*: entities representing pedestrians from the public that happen to present at the tracking scene and consequently get involved in the tracking process. Agents can either be masters or slaves, see section 4.4.

For obvious reasons, trackers should not be physically chasing the targets; instead, they recruit tracking agents who eventually become part of the tracking network. Briefly, the recruitment process proceeds in two steps (this process is formalised in section 4.5):

1. *Range Search*: first, the recruiter (who is either a tracker or a master agent) randomly searches its range for entities.
2. *Recruitment*: once the recruiter finds some entities, it checks whether these entities are suitable for recruitment, and if so, it recruits them.

We assume that the tracker can identify the target, typically in the form of a nominally unique address, and that, initially, the target is in the tracker’s range. We also assume all entities involved in the tracking process support similar communication range, which is important to maintain consistent communication. Once the tracker recruits the agents, the tracker triggers them to form *piconets*; a piconet is a small ad hoc network consisting of one master and up to 7 (active) slaves<sup>1</sup>. The farther the target moves, the more piconets are formed. While discussing the algorithm, we will be introducing two types of piconets, *tracking piconets* and *connecting piconets*.

- The tracking piconet is responsible for the main tracking task (i.e., locating the target and sending location updates to the tracker).
- The connecting piconets are responsible for maintaining connectivity throughout the network by providing delivery links from the tracking piconet to the tracker and maintaining the delivery of the tracking updates.

Figure 4.1 illustrates a typical piconet interconnection scenario; more information about these piconets is given in section 4.4. The tracker only recruits the agents of the first piconet, subsequent recruitments are then handled by the masters. If an agent (master or slave) is no longer needed, it is retired; algorithms for the recruitment and retirement of agents are provided in section 4.6.1. Moreover, the tracker maintains a tracking table  $\mathcal{T}$  containing records of the target’s movements history which are being updated periodically by the tracking piconet. Table 4.1 is a sample  $\mathcal{T}$  (addresses in table 4.1 were generated randomly and do not correspond to real addresses).

Date	Time	Target	Location
10-11-08	13:04:22	10-33-af-b2-00-c5	51.3833 -2.3500
10-11-08	13:09:01	1f-04-e6-20-b0-10	51.3833 -2.3533
10-11-08	13:11:59	10-33-af-b2-00-c5	51.3838 -2.3604
...	...	...	...

Table 4.1: Tracking Table snapshot

<sup>1</sup>A piconet may consist of more than 7 slaves, but only up to 7 slaves can be active ant any given time, while the rest are said to be passive. This, however, will introduce unnecessary overhead.

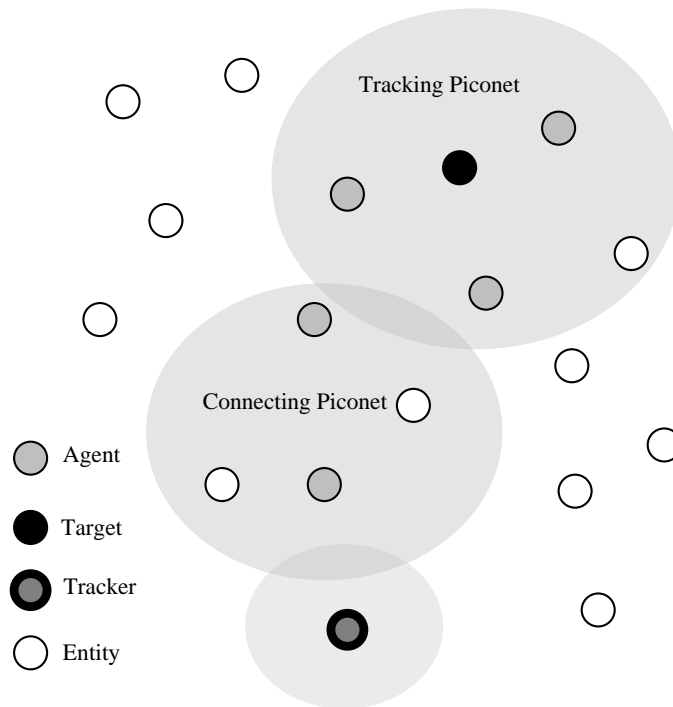


Figure 4.1: A sample single-target single-tracker tracking network consisting of one tracking piconet and one connecting piconet

We assume that the recruiter is able to communicate with the agent and install<sup>2</sup> tracking software. Our tracking system is based on two software components installed in the agents at recruitment, these are:

1. *Probe-SW*: used to locate the suitable entities for recruitment.
2. *T-SW* (tracking SW): responsible for various management and tracking tasks.

In addition, we also introduce a number of management *signals* and messages that are occasionally exchanged among the agents to trigger various tasks. These signalling messages are discussed throughout the remainder of the chapter and are summarised in table 4.2. This table should be treated as a reference and the reader can come back to it later when necessary. Finally, we assume that we have knowledge of the initial location of the target. This is plausible because in order for the tracker to initiate the tracking process, the target has to be (initially) in the tracker’s range, and since we are considering a short-range communication, the tracker will, very likely, be able to estimate the target’s location (even visually).

<sup>2</sup>The software can be distributed either by cooperating with the carrier or through active propagation mechanisms that may or may not involve the consent of the agent depending on legal requirements, which may significantly vary from a country to another, see sections 3.2.4 and 4.8.

Signal/message	Description
form_pico signal	a signal sent to an agent triggering it to form a new piconet for which it will act as a master
self-destruction signal	a signal sent to an entity to retire it
alive messages	exchanged among agents to maintain the network
sensing request	sets the agents to the temp mode where they constantly search for genuine trackers
authentication request	used for authentication purposes between agents and trackers; see section 4.7.1
discover message	broadcasted by an agent to discover its neighbouring agents
neighbourList signal	exchanged among agents to request lists of each other's neighbouring agents
mastership signal	sent by the backup master to the slaves of a piconet to forcefully take over the mastership of the piconet
target location update	messages containing information about the current location of the target (see section 4.1), these messages are sent by the tracking master(s) to the tracker(s)

Table 4.2: Signals and messages exchanged among the agents and the trackers

### 4.3 Online Pedestrian Localisation

As discussed extensively in chapter 2, conventional localisation techniques in wireless networks are based on measuring signal parameters such as: Received Signal Strength (RSS), Time of Arrival (TOA), Time Difference of Arrival (TDOA), and Angle of Arrival (AOA). Since TOA usually requires a fully synchronised network, which is not preserved in our ad hoc environment, and TDOA may involve interacting with the target, which, clearly, defeats the passivity of the tracking process, both TOA and TDOA are not suitable for our algorithm. Similarly, measuring AOA is often difficult as it requires the agents to be equipped with specialised antenna, which we cannot, realistically, assume. This leaves RSS to be the only directly suitable method for our purpose, which can be used in both passive and active tracking. In this section we will describe the localisation of a single target, this can be generalised to the localisation of multiple targets and will later be used in sections 4.6 and 4.7 for single and multiple targets tracking.

### 4.3.1 Passive Localisation

In a passive setting, the target is localised in three steps: first, the recruited agents are (actively) localised, then the distances between those agents and the target are estimated by means of RSS, and finally the target’s location is estimated via trilateration.

**Step 1: Localising the Agents.** Initially, in addition to the location of the target, we assume that the tracker can also estimate the location of at least two of the recruited agents. Such assumption is plausible for a short-range communication since, in the worst case scenario, the tracker can visually estimate the locations of the agents. Alternatively, these two agents can be associates of the tracker to help with the initial localisation process after which they may depart. Along with the tracker, these two agents can localise a third agent by means of trilateration (see below). In fact, any entity can be localised as long as there are at least three other agents with known locations in its range. It is important that the three agents localising the target have themselves been already localised, other agents (if there is any) do not have to.

It is also possible to use active methods based on, e.g., TOA/TDOA, to localise these agents since we are not concerned about passivity between agents; see section 4.3.2. Furthermore, if the agents are equipped with GPS (which is very common today), the location of the agents can be obtained even more conveniently, having that the agents are located in an environment where GPS signals are available (e.g., outdoor).

**Step 2: Measuring Distances.** Based on the discussion at the beginning of this section, we use RSS to estimate the distance between the agents and the target. In long-range communication, RSS measurements may be affected by a number of factors. While moving objects such as vehicles and pedestrians may introduce random errors that can be eliminated (or mitigated) by averaging the measurements, the static non-moving objects, such as buildings, will introduce systematic errors that are difficult to filter without careful modelling of the environment [96] (see section 2.5). Fortunately, in short-range communication, the noise due to static objects is, on average, acceptable as the communication covers limited areas and does not travel long distances. Thus, in our localisation algorithm, we model the environment by adopting the free-space radio propagation model, which abstracts away both the random and the systematic errors. Free-space model is a basic radio propagation model for measuring RSS that assumes a clear line-of-sight (LOS) path between the transmitter and the receiver, which, in most short-range communicate scenarios, may be preserved. Hence RSS is calculated as follows:

$$\frac{P_{received}}{P_{transmitted}} = \frac{A_{receiver} \cdot A_{transmitter}}{d^2 \cdot \lambda^2}$$

This equation is known as Friis equation [43], where  $P_{received}$  and  $P_{transmitted}$  are the received and transmitted power, respectively,  $A_{receiver}$  and  $A_{transmitter}$  are the effective area of the receiving and transmitting antenna, respectively,  $d$  is the distance between the transmitter and the receiver, and  $\lambda$  is the wavelength. Assuming that the target uses an isotropic antenna<sup>3</sup>, the power radiating from the target is calculated as follows:

$$P_{received} = \frac{P_{transmitted} \cdot A_{receiver} \cdot A_{transmitter}}{4\pi \cdot d^2}$$

hence, the distance between the transmitter and the receiver is:

$$d = \sqrt{\frac{P_{transmitted} \cdot A_{receiver} \cdot A_{transmitter}}{4\pi \cdot P_{received}}}$$

Measurement filtering and multi-sensor data fusion (fusing measurements from multiple sensors) [41] can also be used to improve the measurement accuracy but with an increased computational complexity that the agents may not be capable of. City radio propagation models, like Young, Okumura, Hata and Lee models [116], may enhance the accuracy of the measurements in some cases, but as discussed in section 2.5, these models were developed based on empirical data collected in real cities primarily targeting the radio propagation properties for long-range communication scenarios and would most likely perform poorly when applied on short-range scenarios. This is primarily why we opt to adopt the free space model, but regardless of the adopted model, in urban and other dense environments, there will necessarily be an unavoidable radio propagation error margin.

Once the distances are estimated, every reference agent sends its respective estimated distance to the master agent and the algorithm proceeds to the final step where the location of the target is estimated.

**Step 3: Trilateration.** As discussed in section 2.2.2, trilateration is a well-known geometric approach for localising a target based on at least 3 *non-collinear* reference points (i.e., the reference points should not all lie on a single line) when localising in 2 dimensions, or at least 4 non-collinear reference points when localising in 3 dimensions. Although in this algorithm we consider 2 dimensions scenarios, this can be easily extended to 3 dimensions. Once the distances between the agents and the target are estimated as described above, the target is localised by calculating the intersection

---

<sup>3</sup>An isotropic antenna is based on the theoretical isotropic radiation concept. The radio waves emitted from this antenna are radiated uniformly in all directions. It is not clear whether such theoretical antenna exists because achieving this behaviour in practice is very difficult, if at all possible, but the concept is useful as a reference model for an ideal antenna to which other (practical) antennas can be compared.

point of the three circles formed by the three reference agents that are centred at the locations of the agents and have radii of the agents' respective estimated distances from the target (as calculated in step 2). Figure 4.2 illustrates how trilateration is performed, where  $(x_i, y_i)$  and  $r_i$  are the centre and radius of the circle  $i$  which corresponds to Agent $_i$ . Knowledge of the centres of these circles (the coordinates of the locations of the reference agents) and their corresponding radii (the distances between the reference agents and the target) is sufficient to calculate the circles' intersection point.

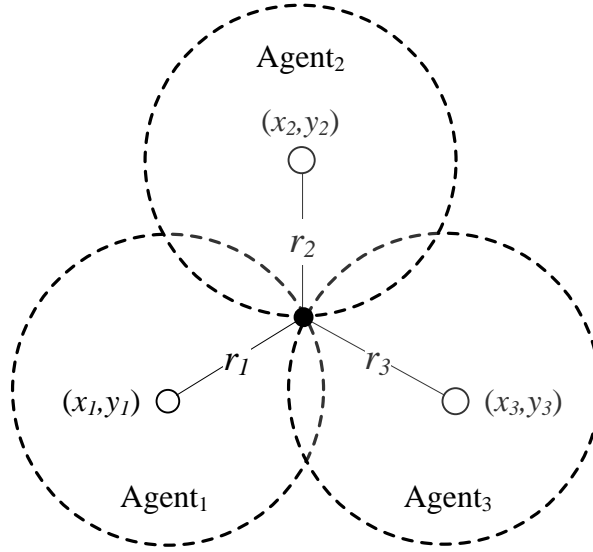


Figure 4.2: Basic Trilateration

The algorithm proceeds as follows: we know that the equation of a circle is  $(x - h)^2 + (y - k)^2 = r^2$  for all points  $(x, y)$  on the circle, where  $(h, k)$  are the coordinates of the centre of the circle and  $r$  is its radius. Based on figure 4.2, the equations of the three circles are:

$$\begin{aligned} (x - x_1)^2 + (y - y_1)^2 &= r_1^2 \\ (x - x_2)^2 + (y - y_2)^2 &= r_2^2 \\ (x - x_3)^2 + (y - y_3)^2 &= r_3^2 \end{aligned}$$

then, by elementary transformations, the coordinates  $(x, y)$  are:

$$x = \frac{fb - ec}{bd - ea} + \epsilon, \quad y = \frac{c -afb + aec}{b^2d - ab} + \epsilon$$

where  $a = 2x_1 - 2x_2$ ,  $b = 2y_1 - 2y_2$ ,  $c = r_2^2 - r_1^2 - x_2^2 + x_1^2 - y_2^2 + y_1^2$ ,  $d = 2x_1 - 2x_3$ ,  $e = 2y_1 - 2y_3$ ,  $f = r_3^2 - r_1^2 - x_3^2 + x_1^2 - y_3^2 + y_1^2$ , and  $\epsilon$  is random variable representing

measurement and radio propagation errors. For the sake of expediency, the circles are considered to be threshold values of cumulative probability distributions (PDFs) incorporating error margins of the RSS measurements. In particular, we assume that the three circles will be aligned perfectly to intersect in one point; However, this may not be the case in practice. That is, it is more likely that the three circles will not intersect in exactly one point, but rather 3 points forming a *circular triangle*. In this case, the target may be localised by finding these three points, treat them as vertices for an ordinary triangle and find its centroid where the target will most likely be. However, this process will require advanced calculations and knowledge of some angles between the reference agents which we assumed to be hard to obtain; detailed discussion of this more accurate (and advanced) trilateration process is presented in section 2.2.2. Algorithm 4.1 illustrates the localisation algorithm.

---

**Algorithm 4.1** Single Target Localisation
 

---

```

1: SET Tracker {identify the tracker}
2: SET Target {identify the target}
3: AgentsCount  $\leftarrow$  3 {at least 3 agents}
4: for  $i = 1$  to  $i \leq$  AgentsCount do
5:   Agents[i].addr  $\leftarrow$  getAddress(Agent[i]) {Step 1: identify agents}
6:   Agent[i].location  $\leftarrow$  getLocation(Agent[i]) {locate the agents}
7: end for
8: for  $i = 1$  to  $i \leq$  AgentsCount do
9:   Agent[i].RSS  $\leftarrow$  measureRSS(Agent[i],Target) {Step 2: measure RSS}
10: end for
    {Step3: Trilateration}
11: Target.location  $\leftarrow$  Tri(Agent[1,2,3].location, Agent[1,2,3].RSS)
12: locationUpdate(Target.location, Tracker) {send location update to the tracker}

```

---

### 4.3.2 Active Localisation

As illustrated in algorithm 4.1, the first step in our localisation algorithm is to localise the agents. Since localising the agents need not be passive, we can use active localisation methods, which are likely to be more accurate. Evidently, time-based parameter measurement techniques, such as TOA and TDOA, usually enjoy higher accuracy margin than the signal strength-based ones. Thus, the distance between agents  $A_1$  and  $A_2$  can be estimated as follows (algorithm 4.2 illustrates this process, which is known as *round-trip time*, see section 2.2.1):

1.  $A_1$  sends a signal to  $A_2$  at time  $t_1$ .
2.  $A_2$  receives the signal and sends an acknowledgement back to  $A_1$ .

3.  $A_1$  receives the acknowledgement at time  $t_2$ .
4. Assuming the radio waves propagate in free space at approximately the speed of light ( $c \approx 3 \times 10^{-8}m/s$ ),  $A_1$  calculates the distance between  $A_2$  and itself using the conventional distance equation:  $d = c \times t$ , where  $t = t_2 - t_1$ .

This procedure can also be used with the target occasionally if passivity is not strictly required at all times.

---

**Algorithm 4.2** Round-trip Time
 

---

```

1: SET  $A_1, A_2$  {identify two agents}
2:  $A_1$ .sendSignal( $A_2$ ) { $A_1$  sends signal to  $A_2$ }
3:  $A_1$ .sendSignal( $A_2$ ).time =  $t_1$  { $A_1$  records the time the signal went out}
4: if  $A_2$ .receiveSignal( $A_1$ ) = TRUE then
5:    $A_2$ .sendAck( $A_1$ ) { $A_2$  sends acknowledgement back to  $A_1$ }
6: end if
7: if  $A_1$ .receiveSignal( $A_2$ ) = TRUE then
8:    $A_1$ .receiveSignal( $A_2$ ).time =  $t_2$  {record the time  $A_1$  receives the ack}
9:   distance( $A_1, A_2$ ) =  $3 \times 10^{-8}(t_2 - t_1)$  {calculate the distance between  $A_1$  and  $A_2$ }
10: end if

```

---

## 4.4 Piconets

As a result to the restrictions that the short-range communication nature of our tracking environment imposes on the agents, the tracking network will eventually be split into several *piconets*. As briefly described in section 4.2, our tracking algorithms utilise two types of piconets, tracking piconets and connecting piconets.

### 4.4.1 Tracking Piconet

The actual localisation of the target takes place in the tracking piconet. The master of the tracking piconet is called *tracking master* (which has a central role in carrying out the tracking process) and the slaves are called *tracking slaves*. To be able to localise the target, at least 3 members of the tracking piconet should be within the target's range (see figure 4.1). Once the target is localised, the tracking master transmits the location updates back to the tracker. The first tracking piconet is formed by the tracker at the initialisation stage of the tracking process, then subsequent tracking piconets are formed on demand. However, a tracking network cannot have two tracking piconets for a single target; hence, once a new tracking piconet is formed, the old one is either torn-down or downgraded to a connecting piconet.

### 4.4.2 Connecting Piconet

The connecting piconet function as a bridge between the tracking piconet and the tracker, connecting the two and handling the transmission of the traffic between them (i.e., tracking updates and acknowledgements). In the connecting piconet, the master is called *connecting master* and the slaves are called *connecting slaves*. Obviously, the further the target moves away from the tracker, the more connecting piconets are formed. Every connecting piconet has at least two *bridge agents* (one of which can be the master) to link the connecting piconet with its neighbours (at the tracker's closest connecting piconet, the tracker is connected to one of these bridge agents). In fact, to connect the tracking network, one agent is sufficient, but this will most likely over-utilise the agent's resources.

## 4.5 Piconet Formation

The *piconet initiator* is the entity that triggers the formation of a piconet. The tracker initiates the very first piconet, which is a tracking piconet, and then sets passively to receive the target location updates; all subsequent piconets are formed by master agents. New piconets are formed in two cases:

- a new tracking piconet is formed if the current tracking piconet can no longer handle the target's movements or if the current tracking piconet broke down,
- a new connecting piconet is formed if an intermediate connecting piconet broke down or as a downgraded tracking piconet.

Below we propose two piconet formation algorithms to cope with any formation demand as outlined above, these are: Formation by Direct Interrogation (FDI), and Formation by Neighbour Interrogation (FNI). Figure 4.3 is a flow chart illustrating the overall mechanics of both algorithms (note that the minimum required number of agents to form a connecting piconet is 2 while it is 3 for a tracking piconet; see figure 4.7). We discuss the formation algorithms below when forming tracking piconets, forming connecting piconets is similar except when forming a connecting piconet the Probe-SW does not check the range of its host for the target.

### 4.5.1 Formation by Direct Interrogation (FDI)

In FDI, the initiator of the piconet searches its range, randomly chooses one of the entities and sends the Probe-SW to it. The Probe-SW then searches the entity's range

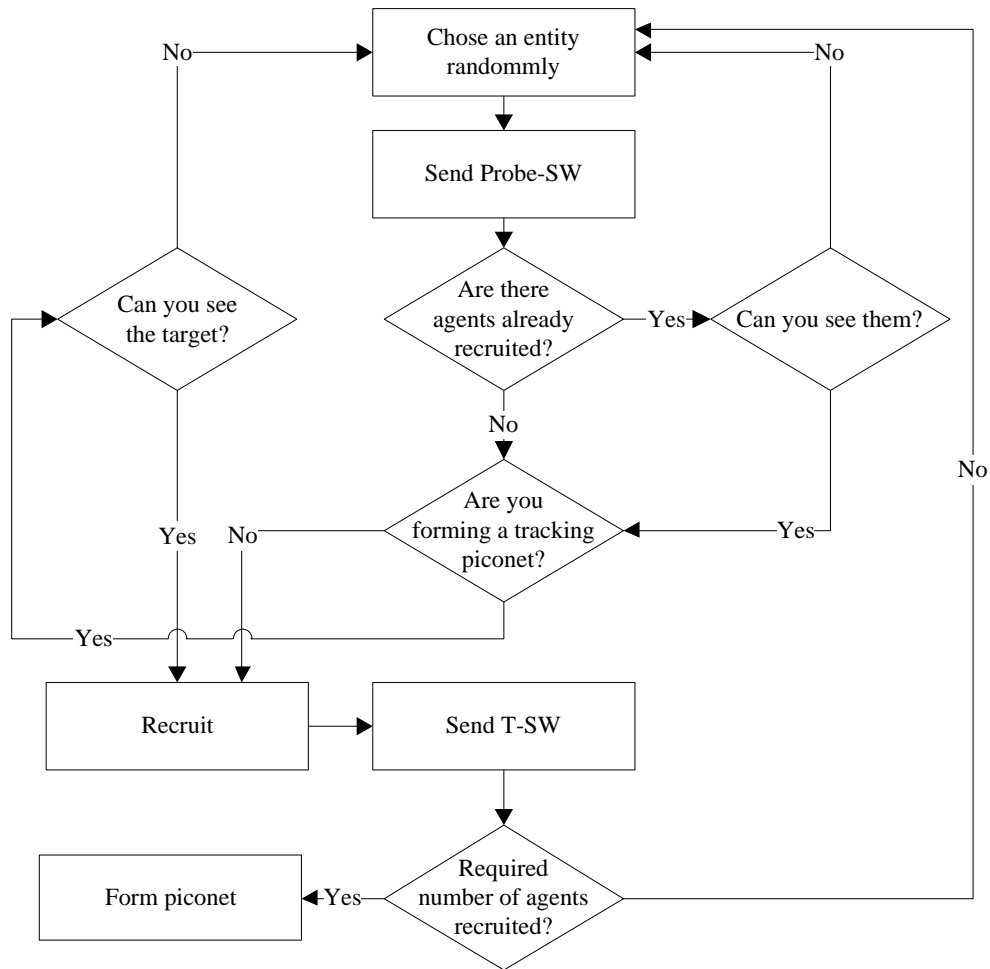


Figure 4.3: Flow chart illustrating the FDI and the FNI algorithms

for the target<sup>4</sup> and reports back to the initiator. If the target was found, the initiator recruits that entity by sending the T-SW to it, otherwise, the initiator discards the entity and searches its own range again to recruit a different one. When the first agent is recruited, the initiator attempts to recruit a second one by repeating the process above but this time it asks the Probe-SW to search the range of the entities it is sent to for both the target and the already recruited agent. To successfully form a piconet, it is important that the recruited agents can see each other as well as seeing the target. When the second agent is recruited, the initiator repeats the same process to recruit a third agent after which the initiator would have recruited the minimum required

<sup>4</sup>Here we describe FDI when forming a tracking piconet, forming a connecting piconet is similar, except that when forming a connecting piconet, the Probe-SW may be triggered to search for any other entity such as the master of an adjacent piconet.

number of agents to be able to localise the target<sup>5</sup> and can now form a tracking piconet by sending the *form\_pico* signal to one of the agents to act as the piconet’s master (for example, the agent with highest ID may be selected for this purpose). The *form\_pico* signal contains the addresses of the already discovered agents which will be the slaves of that piconet; in this case, the master does not have to rediscover its slaves because the agents have been recruited in such a way that guarantees that all of them can see each other, as well as the target. Algorithm 4.3 illustrates the FDI algorithm.

---

**Algorithm 4.3** Formation by Direct Interrogation (FDI)

---

```

1: SET Initiator {identify initiator of the piconet}
2: SET Target {identify the target}
3: SET MIN {minimum number of agents to form a piconet}
4: SET Agents[MIN] {create an array of size MIN, index starts at 1}
5: for  $i = 1$  to  $i \leq \text{MIN}$  do
6:   Entity[ $i$ ]  $\leftarrow$  SearchRange(Initiator)
7:   for  $j = 1$  to  $j \leq \text{MIN}$  do
8:     Entity[ $i$ ].Suitable  $\leftarrow$  SendProbe-SW(Entity[ $i$ ],Target, Agents[ $j$ ])
           {return TRUE if the Target and all recruited agents are in Entity[ $i$ ]’s range}
9:   end for
10:  if Entity[ $i$ ].Suitable = TRUE then
11:    sendT-SW(Entity[ $i$ ])
12:    Agent[ $i$ ]  $\leftarrow$  Entity[ $i$ ] {Recruit Entity[ $i$ ]}
13:  end if
14: end for
15: Master  $\leftarrow$  recruitMaster(Agent[ $x$ ]) {select a master,  $x \in \{1, 2, \dots, \text{MIN}\}$ }
16: sendPico_form(Master) {form the piconet}
17: for  $i = 1$  to  $i \leq \text{MIN}-1$  do
18:   if Agent[ $i$ ]  $\neq$  Master then
19:     recruitSlave(Agent[ $i$ ]) {recruit other agents as slaves}
20:   end if
21: end for

```

---

#### 4.5.2 Formation by Neighbour Interrogation (FNI)

Another way to form a piconet is by modifying the Probe-SW to discover the neighbours of the entity it is sent to and returns a list of those neighbours to the initiator. The initiator then analyses this information and decides whether the corresponding entity is suitable for recruitment (i.e., whether the target is listed as one of its neighbours). If that agent is recruited, the initiator randomly chooses one of that agent’s neighbours

---

<sup>5</sup>For better localisation accuracy and fault tolerance, the initiator may wish to recruit more agents; see section 4.7 and figure 4.7 which illustrates the minimum and maximum number of agents that can be recruited in a single piconet.

and sends the Probe-SW to it to retrieve the neighbours of the latter and repeats the process until the minimum required number of agents are recruited (i.e., three agents for a tracking piconet and two agents for a connecting piconet). Algorithm 4.4 illustrates the FNI piconet formation algorithm.

---

**Algorithm 4.4** Formation by Neighbour Interrogation (FNI)

---

```

1: SET Initiator {identify initiator of the piconet}
2: SET Target {identify the target}
3: SET MIN {minimum number of agents to form a piconet}
4: SET Agents[MIN] {create an array of size MIN, index starts at 1}
5: for  $i = 1$  to  $i \leq \text{MIN}$  do
6:   Entity[ $i$ ]  $\leftarrow$  searchRange(Initiator) {select an entity at random}
7:   NeighbourList  $\leftarrow$  getNeighbourList(Entity[ $i$ ])
8:   for  $j = 1$  to  $j \leq \text{sizeof}(\text{NeighbourList})$  do
9:     for  $x = 1$  to  $x \leq \text{MIN}$  do
10:      Entity[ $i$ ].Suitable  $\leftarrow$  sendProbe-SW(NeighbourList[ $j$ ], Target, Agent[ $x$ ])
      {return TRUE if the Target and all agents are neighbours to Entity[ $i$ ]}
11:    end for
12:    if Entity[ $i$ ].Suitable = TRUE then
13:      sendT-SW(Entity[ $i$ ])
14:      Agent[ $i$ ]  $\leftarrow$  Entity[ $i$ ] {Recruit Entity[ $i$ ]}
15:      break {Recruit another agent}
16:    else
17:      continue {loop and try different neighbour}
18:    end if
19:  end for
20: end for

```

---

## 4.6 Basic Pedestrian Tracking

Since the target's movement is not restricted, agents will need to be dynamically recruited and retired. However, this requirement poses several challenges while forming and tearing-down communication between the agents. In this section we consider such issues in a single-tracker single-target scenario; advanced algorithmic treatment for the multi-target multi-trackers scenarios is provided later in section 4.7.

### 4.6.1 Agent Recruitment and Retirement

As briefly discussed in section 4.2, the tracker initiates the tracking process by broadcasting a specialised software, Probe-SW, to every entity in its range. The Probe-SW first evaluates its host entity for suitability of requirement (e.g., whether the target is

in range) and replies back to the tracker. The tracker then decides whether to recruit the entity. Recruitment at this stage is triggered by the tracker and is merely a matter of sending another software, T-SW, to the suitable entities. However, recruiting and retiring agents in subsequent stages are processed more systematically; in most cases, recruitment is handled by the masters of the corresponding piconets. Retirement, on the other hand, are triggered for

- entities failing the Probe-SW initial evaluation, and
- agents departing from their piconets.

An agent is retired by receiving a *self-destruction* signal from its master (if it is a slave), the new master (if it is a master departing from its piconet), or the recruiter (if it was an entity failing the Probe-SW evaluation). This signal will trigger the self-destruction process, where Probe-SW and T-SW (if present) are removed from the host entity.

**Departing Slave.** Since agents are mobile entities, they will eventually depart from the tracking network. Thus, it is crucial to detect their departure as soon as possible to allow sufficient time for retiring them and arrange for recruiting replacements. All agents (masters and slaves) regularly exchange *alive* messages to maintain the piconets. We do not define how often these messages should be exchanged, but we note that since we are modelling human movements, rapid exchange rate may be unnecessary. The purpose of the alive messages is to assure the reachability of the entities and are sent from the master to all its slaves repeatedly. Once the slaves receive these messages, acknowledgments should be sent back. If the master does not receive several consecutive acknowledgements from a particular slave, it retires it and recruits another. Algorithm 4.5 illustrates how the departure of a slave is handled.

### 4.6.2 Simulation Results

We used Network Simulator 2 (NS-2) to evaluate the proposed algorithms. Figure 4.4 presents the simulation results for a scenario consisting of 18 entities running for 30 simulation minutes. The simulated entities were being constantly recruited/retired by a single tracker while tracking a single target. The figure shows the difference in distance between the actual location of the target (obtained from NS-2 built-in location function) and our algorithm's estimated location. We used the following formula to calculate the difference between the two locations:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

**Algorithm 4.5** Detection of a Departing Slave

---

```

1: SET SlaveCount {how many slaves}
2: SET Master {identify the master}
3: SET Slave[SlaveCount] {identify the slaves}
4: SET max_missed {max. number of missed acknowledgements}
5: while Tracking is active do
6:   for  $i = 1$  to  $i \leq$  SlaveCount do
7:     sendAlive(Master  $\rightarrow$  Slave[ $i$ ]) {exchange alive messages}
8:     if Ack = received then
9:       continue {if an acknowledgement is received then loop again}
10:    else if Ack = not_received then
11:      Slave[ $i$ ].missed_ack =+ 1 {count how many missed acknowledgements}
12:      if Slave[ $i$ ].missed_ack  $\geq$  max_missed then
13:        retire(Slave[ $i$ ]) {retire the slave if it exceeds max. no. of misses}
14:        recruit_new();
15:      end if
16:      continue {otherwise, keep exchanging alive messages}
17:    end if
18:  end for
19: end while

```

---

where  $(x_1, y_1)$  and  $(x_2, y_2)$  are the coordinates of the target's actual and estimated locations, respectively. The average difference between the actual and estimated locations was found to be 0.54 meters, which is, in most application, acceptable. Typical human movement in common scenarios (e.g., a target moving through a relatively crowded area) was considered when selecting the simulation parameters to simulate the target and agents where both used the random waypoint mobility model<sup>6</sup> and a speed of up to 1 m/s within an area of 250 m<sup>2</sup>. Furthermore, due to the target's limited speed, we localise the target every 5 seconds where it would most likely have made a movement worth noting. The results in figure 4.4 are based on a *free space* radio propagation model in a 2 dimensions scenario.

## 4.7 Advanced Pedestrian Tracking

Generally, tracking can be handled by single or multiple trackers to track single or multiple targets (while assuming that both the trackers and the targets are mobile entities). This suggests four possible scenarios:

- Scenario 1: Single Tracker tracking Single Target.

---

<sup>6</sup>See section 3.4.1 for description about random waypoint mobility model and other models.

Simulation software	NS-2
Simulation area	250m <sup>2</sup>
Simulation duration	1,800 seconds
Simulation runs	20
Node max. speed	10 m/s
Mobility Model	Random waypoint
Communication technology	IEEE 802.11
Communication range	10m
Radio propagation model	Free space

Table 4.3: Simulation configurations

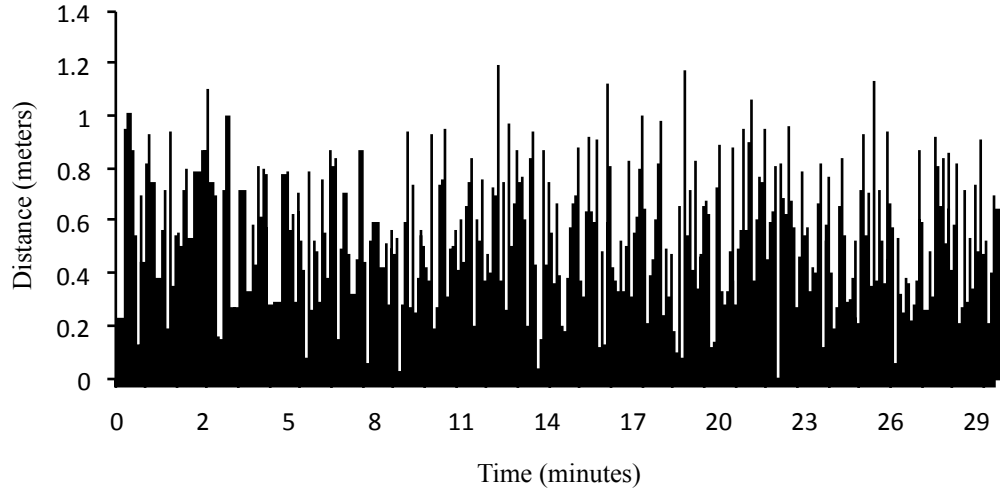


Figure 4.4: Difference in meters between the actual and estimated target locations during 30 simulation minutes

- Scenario 2: Multiple Trackers tracking Single Target.
- Scenario 3: Single Tracker tracking Multiple Targets.
- Scenario 4: Multiple Trackers tracking Multiple Targets.

Scenario 1 is the most basic and is the basis of all other scenario; scenario 1 was thoroughly covered in section 4.6. In scenario 2, we introduce *Handover Tracking*, which involves tracking a target by cooperatively handing over the tracking process between multiple trackers as necessary (see section 4.7.2). In scenario 3, we introduce *Virtual Tracking*, a technique for tracking more than one target (see section 4.7.1). Finally,

tracking in scenario 4 is based on all the above techniques and is the most complex. In all scenarios, we assume that trackers have knowledge of the targets' addresses, which are passed to all recruited agents along with the addresses of all trackers. We also assume that trackers possess a secret key which they share with all recruited agents for lightweight authentication and integrity check as described in section 4.7.1.

#### 4.7.1 Tracking with Multiple Trackers

When target(s) are being tracked by multiple trackers, we assume that a private secure medium linking the trackers is provided (e.g., secure federal network), allowing them to periodically synchronise the tracking information. Clearly, it is usually the case that such network exists among law enforcement officers. In this scenario, we introduce a framework called *handover tracking*, where the trackers cooperatively track the target(s). Usually, the tracking network is handled by a single tracker at any give time, even if multiple trackers are concurrently available (scenario 4 above is the exception, see section 4.7.2). However, if the tracking network breaks down, either a tracking handover to another tracker occurs or a temporary tracker is elected until a genuine tracker becomes available (this is discussed in section 4.7.4, also see algorithm 4.9).

Ideally, the tracker acknowledges receiving every *target location update* sent by the tracking master. If the tracking master does not receive such acknowledgement after 3-4 tracking updates, it assumes a communication failure has occurred. The tracking master then senses its own range, if one of the genuine trackers is found, it authenticates it (see below) and starts transmitting the location updates directly to it. Otherwise, the tracking master elects itself as a *temporary tracker*, creates a temporary tracking table  $\mathcal{T}_{temp}$ , and breaks the whole tracking network to start a new one from scratch acting as a tracker to recruit agents and form a new tracking network. When the temporary tracker creates this new tracking network, it sets it to a *temp mode*. In this mode, the temporary tracker, beside providing a list of all the genuine trackers and targets to the agents it recruits, it also includes a sensing request, which basically requires all recruited agents to periodically sense their ranges for any of the genuine trackers. If a genuine tracker is detected by a particular piconet (either by its master or slaves), the detecting piconet authenticates the genuine tracker and requests the temporary tracking table  $\mathcal{T}_{temp}$  from the temporary tracker for the genuine tracker. When the genuine tracker receives  $\mathcal{T}_{temp}$ , it merges it with its perviously synchronised copy of the (outdated) tracking table  $\mathcal{T}$ . Finally, the detecting piconet diverts all location updates to the (new) tracker and retires any redundant piconet. This process is formally discussed in section 4.7.4 and illustrated in algorithm 4.9.

**Secure Handover.** Since addresses of both trackers and agents can easily be spoofed during tracking handover, it is important to enforce an authentication process between trackers and agents. At the time of recruitment, all agents are provided with a secret key  $s\_key$ , that is also shared among all trackers. We propose a Secure Handover (SH) scheme, which uses  $s\_key$  to enforce a 3-way handshake authentication process similar to CHAP (Challenge Handshake Authentication Protocol) [119] while involving random numbers (i.e., nonce<sup>7</sup>) to provide extra layer of protection against replay attacks. Figure 4.5 depicts how mutual authentication under the SH scheme takes place between two entities (in this case, between a genuine tracker and the master of a detecting piconet, but of course it can be used between any other entities); algorithm 4.6 provides a pseudocode for our SH scheme.

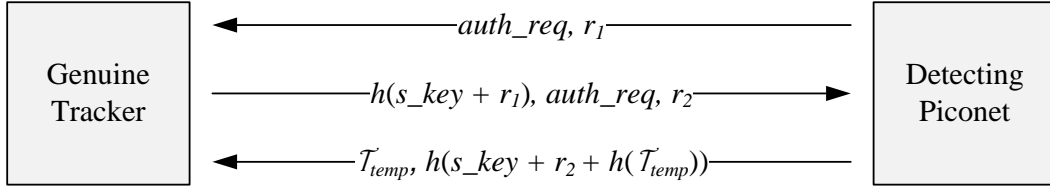


Figure 4.5: Mutual Authentication

---

**Algorithm 4.6** Secure Handover (SH)
 

---

- 1:  $E_1, E_2$  {identify two entities that need to be mutually authenticated}
  - 2:  $auth\_req_1 \leftarrow E_1.GenAuthReq$  {generate an authentication request}
  - 3:  $r_1 \leftarrow E_1.GenRand$  {generate random number}
  - 4:  $E_1.send((auth\_req_1, r_1) \rightarrow E_2)$  {send  $auth\_req_1$  and  $r_1$  to  $E_2$ }
  - 5: **if**  $E_2.received(auth\_req_1, r_1)$  **then** {if an authentication request is received}
  - 6:    $h(s\_key + r_1) \leftarrow E_2.GenHash$
  - 7:    $auth\_req_2 \leftarrow E_2.GenAuthReq$
  - 8:    $r_2 \leftarrow E_2.GenRand$  {generate second random number}
  - 9:    $E_2.send((h(s\_key + r_1), auth\_req_2, r_2) \rightarrow E_1)$
  - 10: **end if**
  - 11: **if**  $E_1.received(h(s\_key + r_1)) = h(s\_key + r_1)$  **then**
  - 12:    $E_1.reqTemp$  {if authenticated, request  $T_{temp}$  from the temp tracker}
  - 13:    $E_1.send((T_{temp}, h(skey + r_2 + h(T_{temp}))) \rightarrow E_2)$
  - 14: **end if**
- 

Once a piconet detects a genuine tracker, the master of that piconet sends an *authentication request* (challenge) to the detected tracker along with a random number  $r_1$ . The tracker then adds  $r_1$  to its copy of  $s\_key$ , hashes it  $h(s\_key + r_1)$ , and sends the hash value back to the detecting piconet along with another authentication request and

<sup>7</sup>A nonce is a random number used only once in a cryptographic setting.

a random number  $r_2$ . Once the response is received, the master of the detecting piconet adds the random number it previously generated ( $r_1$ ) to its own copy of  $s\_key$ , hashes it and compares the result to the hash value it received from the tracker, if they match, the tracker is authenticated and the temporary tracking table  $\mathcal{T}_{temp}$  is requested from the temporary tracker for the newly authenticated tracker (similar authentication process can take place between the master of the detecting piconet and the temporary tracker). The  $\mathcal{T}_{temp}$  is then hashed  $h(\mathcal{T}_{temp})$ , and that hash is further hashed after adding  $s\_key$  and the tracker's random number to it,  $h(s\_key + r_2 + h(\mathcal{T}_{temp}))$ . This final hash is then sent to the tracker along with the actual  $\mathcal{T}_{temp}$ . Once received by the tracker, it tests the integrity of the  $\mathcal{T}_{temp}$  (that it has not been modified in transit) by calculating  $h(s\_key + r_2, h(\mathcal{T}_{temp}))$  and making sure that it matches the hash value it just received; this resembles a standard hash function based MAC (Message Authentication Code) algorithm. Clearly, this authentication scheme resists man-in-the-middle attack since no attacker can act as a man in the middle (pretending to be an agent for the tracker and a tracker for the agent) unless they possess knowledge of  $s\_key$ . This scheme also resists replay attacks. We consider two replay attacks:

1. when an attacker observing the authentication session captures the authentication information (i.e., the hash value) en-route to be maliciously used later, and
2. when an agent, either active or retired, is compromised.

In scenario (1), an attacker cannot use the hash value of a particular session in another since every session adds a random number (which is different for different sessions) to the shared  $s\_key$  before hashing it; note that it is important to include a random number in this case, otherwise the hash of  $s\_key$  alone will always be the same. In scenario (2), we suggest enforcing a regular change of  $s\_key$ . Obviously, since both trackers and agents should have a synchronised copy of the  $s\_key$ , no  $s\_key$  change takes place during temporary tracking. Thus, we assume the existence of a key management mechanism/protocol, but the details of such keying mechanism (key generation, distribution and revocation) are beyond the scope of this thesis.

Clearly, Secure Handover (SH) scheme as described above, is based on symmetric cryptography (where multiple parties share the same key), which allows the agents to claim to be trackers, and this becomes problematic if an agent is compromised. An obvious solution would be to use public key cryptography instead, where every entity has two pairs of keys, a public and a private one, but then in order for one entity to authenticate a tracker it has to have access to its public key. This means that all public keys associated to the trackers have to be supplied to the agents at recruitment (compared to the single  $s\_key$  above), and similarly all the public keys of

the agents have to be automatically supplied and synchronised among the trackers if mutual authentication is required<sup>8</sup>. The list of the agent's keys held by the trackers have to also be carefully updated to reflect retirements/recruitments, but we do not delve into the details of this process here.

### 4.7.2 Tracking Multiple Targets

When dealing with multiple targets, we introduce *Virtual Tracking*, which entails creating a separate *Virtual Tracking Network* (VTN) for every target. Each VTN consists of virtual connecting and tracking piconets which in turn made up of virtual agents. Every virtual piconet is identified by a Virtual Piconet ID (VPID), and every agent is identified by a Virtual Agent ID (VAID). Physical piconets and physical agents can have multiple VAID/VPID and be part of multiple VTN's, as illustrated in figure 4.6. The general formate of VPID and VAID is xxyy, where:

- in VPID, xx specifies the target that the corresponding VTN is tracking, and yy is the piconet ID which is unique within the corresponding VTN,
- in VAID, xx is the agent ID which is unique within the agent's virtual piconet, and yy is the piconet ID to which the agent belongs, which is unique within the corresponding VTN.

This format allows for tracking up to 100 targets; in cases where more than 100 targets need to be tracked, xx and yy are expanded as necessary. In multi-tracker multi-target scenario, each VTN is treated as a separate tracking network that may be handled by single or multiple trackers; see algorithm 4.7.

### 4.7.3 Fault Tolerance

In connecting piconets, only two agents are required<sup>9</sup> but two additional agents are also recruited as backups and paired with the original agents. Backup agents accompany the original agents and both should have the same neighbouring agents (unrecruited neighbours do not have to match) so backup agents can seamlessly take over should the agents they are paired with fail. Similar orientation applies for the minimum three agents required to form the tracking piconets. Figure 4.7 illustrates the minimum and maximum number of agents in the connecting and tracking piconets.

<sup>8</sup>This situation becomes even more complicated when we need to add or remove trackers.

<sup>9</sup>This is the bare minimum, but connecting piconets can consist of more than 2 agents. While having larger connecting piconets will most likely reduce the number of the connecting piconets, it will also increase the management overhead on the masters of those piconets.

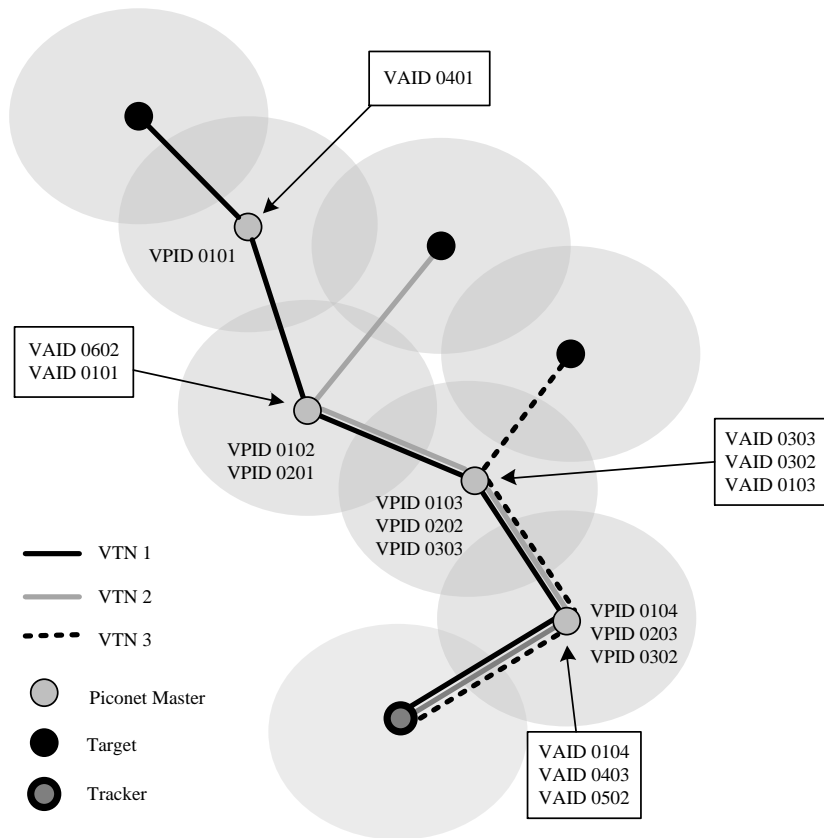


Figure 4.6: Virtual Tracking

All recruited agents are discoverable; that is, every agent can discover its neighbouring agents even if they belong to different piconets. Basically, for an agent to discover all the recruited agents in its range, it broadcasts a *discover* message which any recruited agent should reply to providing its ID (or VAID) and its Piconet ID (or VPID). Within a piconet, agents maintain connectivity by periodically sending *alive* messages to each other—alive messages are exchanged between masters and slaves, but not among slaves; this way, a master can detect a failed slave, and vice versa. Every slave/backup-slave pair is monitored by the master of the corresponding piconet by periodically requiring both the slaves and their backup-slaves to provide lists of their neighbouring agents to confirm that the two lists for a single pair match; if they do not, the backup slave of that pair is retired and the master of that piconet searches for a replacement<sup>10</sup>. All slaves (and backup slaves) send a periodic *alive* signal to the master, thus any failure is immediately detected by the master. If a slave failed, its corresponding backup slave takes over and a replacement backup slave is recruited.

<sup>10</sup>In fact, it is only important that all the neighbouring agents of an agent are also the neighbouring agents of the backup agents, but not vice versa.

**Algorithm 4.7** Multi-Trackers Multi-Targets Tracking

---

```

1: SET GenTrackers {List of genuine trackers}
2: SET Targets {List of targets}
3: SET Agents {List of agents}
4: SET TrackingMaster {identify the master tracker}
5: VTN[sizeof(Targets)] {a VTN for every target}
6: ActiveTracker  $\leftarrow$  GenTrackers[1] {choose an initial tracker}
7: TrackingTable  $\leftarrow$  createTrackingTable(ActiveTracker)
8: for  $i = 1$  to  $i \leq$  sizeof(Targets) do
9:   VTN[ $i$ ]  $\leftarrow$  createVTN(Targets[ $i$ ]) {create separate VTN for each target}
10: end for
11: while ActiveTracker = TRUE do {if a genuine tracker is still in range, continue
    tracking normally}
12:   locationUpdate(TrackingMaster  $\rightarrow$  ActiveTracker)
13:   sync(ActiveTracker, GenTrackers, TrackingTable) {Sync tracking table with other
    trackers}
14:   if ActiveTracker = FALSE then
15:     detectTracker(TrackingMaster, Agents, VTN, GenTrackers)
    {if the tracker becomes unavailable, call the detectTracker procedure}
16:   end if
17: end while

```

---

**Procedure 4.8** Detect Tracker Procedure

---

```

1: detectTracker(TrackingMaster, Agents, VTN, GenTrackers)
   {check if there is another genuine tracker in range}
2: for  $i = 1$  to sizeof(VTN) do
3:   for  $j = 1$  to sizeof(GenTrackers) do {first, check the tracking master's range}
4:     if searchRange(TrackingMaster[ $i$ ], GenTrackers[ $j$ ]) = TRUE then
5:       ActiveTracker[ $i$ ]  $\leftarrow$  GenTrackers[ $j$ ]
6:       ActiveTracker = TRUE
7:       Exit {return to the calling algorithm}
8:     end if
9:     for  $k = 1$  to sizeof(Agents) do {then, check other agent's ranges}
10:      if searchRange(Agent[ $k$ ], GenTrackers[ $j$ ]) = TRUE then
11:        ActiveTracker[ $i$ ]  $\leftarrow$  GenTrackers[ $j$ ]
12:        ActiveTracker = TRUE
13:        Exit {return to the calling algorithm}
14:      end if
15:    end for
16:  end for
   {if no genuine tracker is found, elect a temporary one}
17:  electTemp(TrackingMaster[ $i$ ]) {see algorithm 4.9}
18: end for

```

---

Similarly, the master also has a backup master which should see all the slaves of the piconet. The master confirms so by periodically sending a *neighbourList* signal to its backup master requesting it to provide a list of its neighbouring agents. The master and the backup master also exchange alive messages. If the backup master did not receive alive messages from the master for a particular period of time, the backup master forcefully takes over the piconet by sending *mastership* signal to the slaves and can be mutually authenticated as described in section 4.7.1.

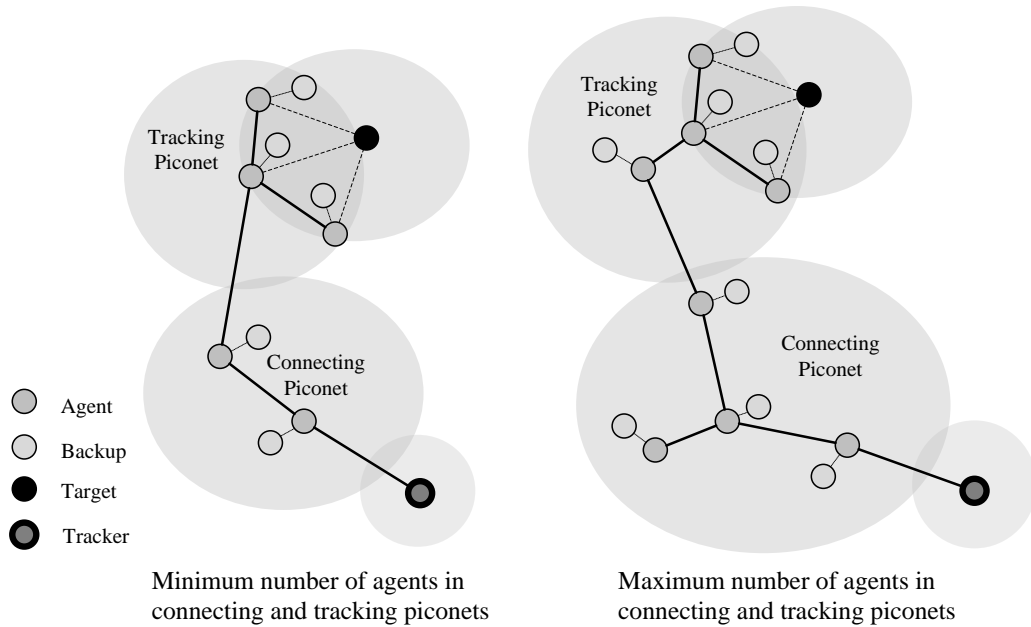


Figure 4.7: Maximum and minimum number of agents forming the piconets

#### 4.7.4 Leader Election

Occasionally, some agents may become unavailable either by physically moving away or by suddenly going offline (e.g., power loss). It is also possible that trackers depart from the tracking network, or the tracking network may break down at any time and cut off the link connecting the tracker. In all these scenarios, replacement agents/trackers have to be recruited/elected. Tracker election process is informally described in section 4.7.1 and is applicable in both single and multiple targets scenarios. This process involves electing a temporary tracker and is activated when the tracking master does not receive 3-4 location updates acknowledgements from the tracker; in this case, the tracking master elects itself as a temporary tracker, see section 4.7.1 for details, and see algorithm 4.9 for a pseudocode of the tracker election algorithm.

**Algorithm 4.9** Temporary Tracker Election

---

```

1: SET GenTrackers {List of genuine trackers}
2: if Failed_ack = 3 then {if 3 location updates aren't acknowledged by the tracker}
3:   TrackingMaster ← TempTracker {elect tracking master tracker as temp tracker}

4:   TrackingNetwork ← break
5:   searchRange(TrackingMaster) {TrackingMaster senses its range}
6:   if GenTracker = TRUE then
7:     divertTraffic(GenTracker)
       {if a genuine tracker is found, then it becomes the tracker}
8:     break
9:   end if
10:  GenTracker = FALSE
11:  TempMode = TRUE {activate temporary mode}
12:  TempTrackingTable ← createTrackingTable(TempTracker)
13:  initiateTracking(TempTracker) {TempTracker initiates the tracking network}
14:  while GenTracker = FALSE do
15:    update(TempTrackingTable)
16:    searchRange(GenTracker) {sense range, can you see any genuine tracker?}
17:    if GenTracker = TRUE then
18:      if authenticate(GenTracker) = TRUE then {authenticate using SH scheme,
        see algorithm 4.5}
19:        GenTracker ← send(TempTrackingTable)
20:        divertTraffic(GenTracker) {divert all location updates to GenTracker}
21:        GenTracker = TRUE
22:      end if
23:      continue {if authentication failed, continue with TempMode}
24:    end if
25:  end while
26: end if

```

---

**4.7.5 Transmission Algorithm**

To improve the passivity of the tracking process, it is important that the transmission of the tracking information does not attract the target's attention. Thus, the tracking information should not always flow over the same route. In this section we propose the transmission algorithm (TA), which is comparable to a routing protocol that does *not* always use the shortest path between nodes—this may slightly tradeoff efficiency. The TA is an optional parameter that only trackers can activate. Without the TA, there is only one route between the tracking piconet and the tracker, but once activated, the algorithm is handled by the tracking master and proceeds in two steps:

1. first, the tracking master creates new routes by recruiting additional agents,

2. then selects which route location updates should take on their way to the tracker.

It is important to create such redundant routes only around the target; these can then merge into a single route down to the tracker (creating redundant routes beyond this point will not improve the passivity of the tracking because, in most cases, it is not observable by the target any way). Algorithm 4.10 formalises these steps, where function  $\text{Recruit}(x, y)$  recruits any entity that is in the range of both  $x$  and  $y$ , and function  $\text{forward}(x \rightarrow y)$  programs  $x$  to forward any traffic it receives to  $y$ .

---

**Algorithm 4.10** Transmission Algorithm (TA)
 

---

```

1: if TA = TRUE then {if the transmission algorithm is activated}
2:   neiPico  $\leftarrow$  getNeighbour(TrackingMaster, ConnectingPico) {get entities that are
   neighbours to both the tracking master and its closest connecting piconet}
3:   SET Agent[MAX] {maximum number of redundant routes}
4:   for  $i = 1$  to  $i \leq$  MAX do
5:     Agent[ $i$ ]  $\leftarrow$  recruit(neiPico) {recruit redundant agents}
6:     forward(Agent[ $i$ ]  $\rightarrow$  neiPico)
7:   end for
8:   Buffer[MAX-1] {set the buffer size}
9:   while TA = TRUE do
10:     $i = 1$ 
11:    while Buffer  $\neq$  Full do
12:      sendUpdates.via(Agent[ $i$ ])
13:      Buffer  $\leftarrow$  append(Agent[ $i$ ]) {keep track of used routes}
14:      increment  $i$ 
15:    end while
16:    purge(Buffer) {reset the buffer once it is full}
17:  end while
18: end if

```

---

**Routes Formation.** To form additional (redundant) routes, the tracking master searches its range for agents that are in both its range and the range of its immediate neighbouring connecting master. Once a suitable agent is found, the tracking master recruits it and configures it to forward any traffic it receives (i.e., tracking updates) to the neighbouring piconet. Every newly recruited agent forms a new route from the tracking piconet to the tracker. Figure 4.8 shows an example where the tracking piconet forms two redundant routes by recruiting two additional agents.

**Route Selection.** The tracking master decides which route a particular tracking update should take by sending that update through the various redundant agents where beyond these agents, all routes merge. In fact, since originally there is only one route

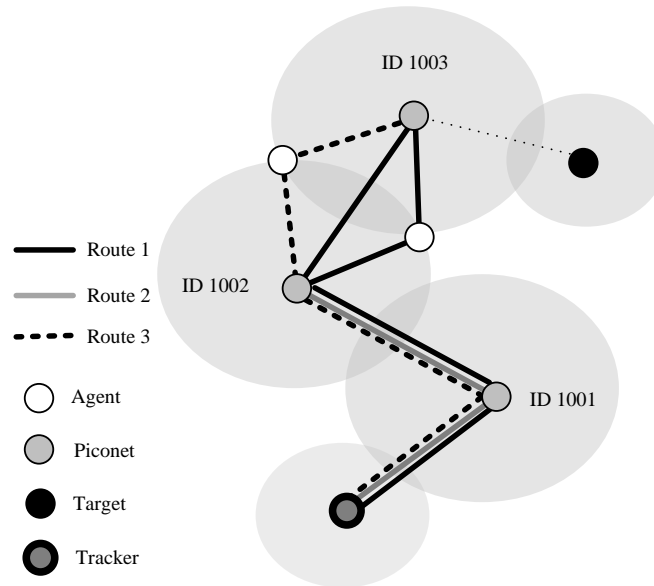


Figure 4.8: The Transmission Algorithm (TA)

between the tracking piconet and the tracker, none of the piconets has to know the exact full route to the tracker. That is, for every connecting piconet, there are two ports (i.e., agents), a receiving port in which traffic enters the piconet and a sending port in which the traffic exits the piconet<sup>11</sup>—any traffic received by any other port, if any exists, is ignored (in the tracking piconet, there is only one port for sending and receiving). When TA is activated, the tracking master maintains a small buffer to remember the addresses of the last few (redundant) agents it sent the tracking updates through, this is important to distribute the tracking updates transmission over as many routes as possible and prevent over-utilising a particular set of routes. That is, if there are  $n$  routes, the tracking master should buffer approximately  $n - 1$  routes and only send the next tracking update on a route that is not listed in that buffer, when the buffer is full, it is purged and the whole process is repeated.

#### 4.7.6 Simulation Results

In section 4.6.2 we simulated the basic localisation/tracking mechanisms, in this section we investigate the effect of mobility models and node density on the tracking process. As discussed earlier, if no tracker is available to receive the tracking updates, the tracking network goes through a temporary tracking period where the tracking master is elected as a temporary tracker until a genuine tracker is detected. However, it is

<sup>11</sup>The ports are reversed when transmitting acknowledgements from the tracker to the tracking piconet (the sending port becomes a receiving port and the receiving port becomes a sending one).

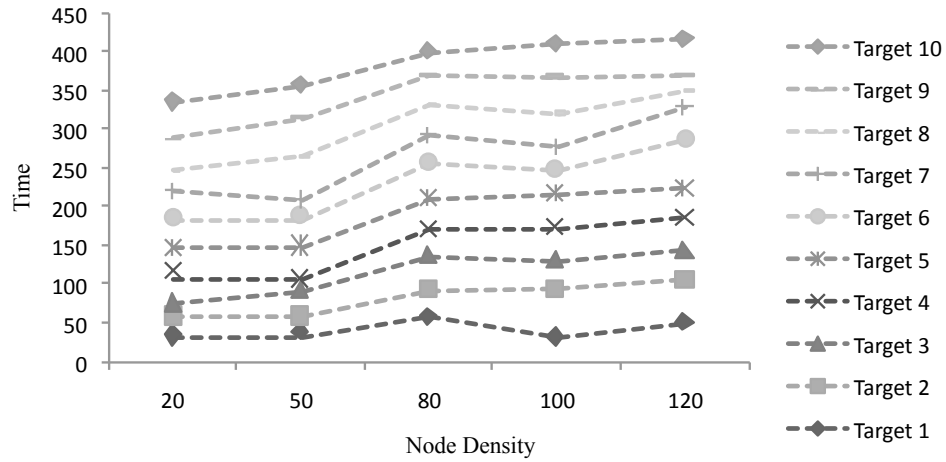


Figure 4.9: Simulation results when adopting the Random Waypoint model

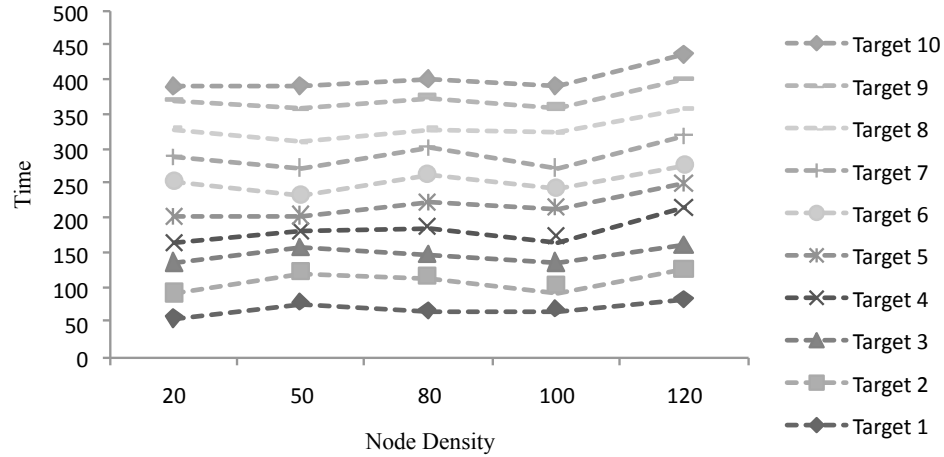


Figure 4.10: Simulation results when adopting the Brownian Walk model

important to investigate how long these *temporary periods* may last because the agents usually have limited resources and may not be able to hold the temporary tracking table  $\mathcal{T}_{temp}$  for long time. Figures 4.9, 4.10, 4.11 show how long, on average, a single temporary period (from losing contact with the trackers until finding them again) lasts in scenarios with different node density (20, 50, 80, 100, 120 nodes) when adopting different mobility models (Random Waypoint, Brownian Walk, Gauss-Markov mobility models, respectively). In these scenarios, the agents have a communication range of 10m and moving over an area of 250 m<sup>2</sup> where 10 targets are being tracked by 3 trackers. Furthermore, to simulate the worst case scenario, we assumed that the agents

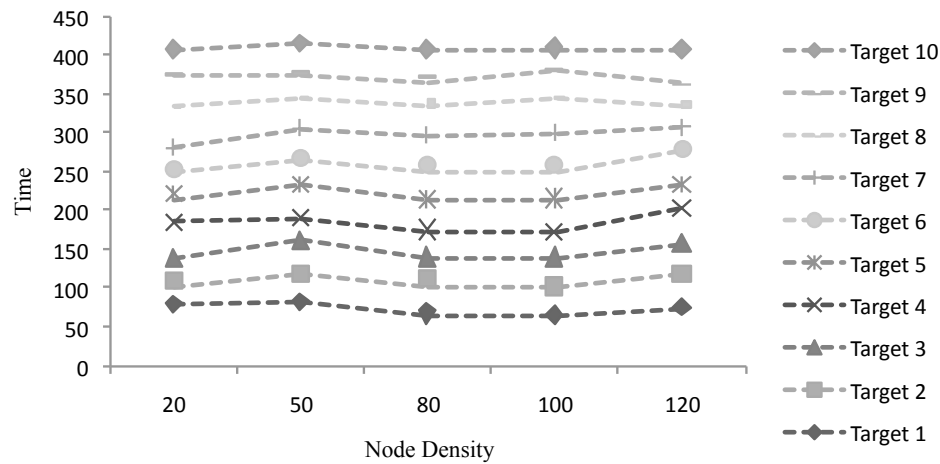


Figure 4.11: Simulation results when adopting the Gauss-Markov model

have limited resources, so agents will only form small tracking networks. Note that figures 4.9, 4.10, 4.11 represent discrete points but are illustrated as *stacked marked lines* to visually show the tenancy of temp time to increase or decrease while increasing the node density (if the figure was plotted normally, the lines will be plotted on top of each other and it will be difficult to visually distinguish between them).

As the figures show, regardless of the mobility model, the node density does (not surprisingly) increase the *temporary tracking* period. However, this is more exemplified in the scenario where the Random Waypoint mobility model (figure 4.9) is adopted, which shows that the choice of the mobility model can affect the simulation results.

## 4.8 Privacy in Online Forensic Tracking

Privacy can become a major issue in online forensic tracking, especially if an agent-based approach is adopted, where, to some extent, privacy implications may be unavoidable. Tracking the target alone, without recruiting agents, might not involve serious privacy violation as this may be warranted, but when external agents are involved in the tracking process, privacy implications arose. Indeed, the recruitment of the agents occur in a random fashion and such agents may not be known to the trackers (law enforcement) beforehand, which means that warrants cannot be issued for them. As discussed in section 3.2.4, some jurisdictions may give power to law enforcement to carry out such course of actions in some situations. However, it is always desirable to preserve the privacy of the agents as much as possible. We thus propose that the following guidelines should be adhered to throughout the tracking process:

Simulation software	NS-2
Simulation area	250m <sup>2</sup>
Simulation duration	1,800 seconds
Simulation runs	10
Node density	20–120
Node max. speed	10m/s
Mobility Model	Random waypoint, Brownian, Gauss-Markov
Communication technology	IEEE 802.11
Communication range	10m
Radio propagation model	Free space

Table 4.4: Simulation configurations

1. The period during which the agents are recruited should be minimised. That is, a particular agent should not stay recruited for more than  $\text{Max}_R$  time, where  $\text{MAX}_R$  is some threshold. Once  $\text{MAX}_R$  expires, the agent should be retired (even if it is still in range within its piconet) and a replacement should be recruited.
2. The Prob-SW and the T-SW should be programmed appropriately to strictly not access the private data of their hosts and only carry out the designated tasks.
3. No record should be kept by the trackers about any recruited agents, and this includes the addresses of the agents, the time during which the agents were recruited and their locations throughout the recruitment.

## 4.9 Summary

In this chapter, we proposed a number of generic passive localisation and tracking algorithms based on short-range communication. The mechanisms presented in this chapter allow intelligence and law enforcement authorities to rapidly and dynamically track targets reasonably efficiently and accurately. The proposed algorithms consider scenarios ranging from single target single tracker tracking, to the multiple target multiple tracker tracking. For scenarios with multiple targets, we introduced *virtual tracking*, where an individual virtual tracking network is created for every target. For scenarios with multiple trackers, we introduced *handover tracking* where the trackers continuously switch the tracking responsibility among themselves based on the target(s) movements. We also proposed a few auxiliary mechanisms to improve the security, passivity and fault tolerance of the algorithms.

## Chapter 5

# Vehicular Forensic Tracking and Motion Prediction

*Vehicular networks have recently attracted significant research interest due to its numerous applications such as those concerning drivers/passengers safety and pleasure. In addition to these civilian applications, vehicular networks can also be used for forensic and law enforcement purposes. In this chapter, we discuss how vehicular networks may possibly contribute to our future crime investigation and prevention. In particular, we propose a high-level algorithmic online vehicular tracking system to passively track a target vehicle. Similar to chapter 4, our vehicular tracking system is based on the dynamic recruitment of the target's neighbouring vehicles as agents. We further propose a mobility prediction algorithm to probabilistically predict the target's near future movement and adjust the tracking process accordingly. Combining both agent-based tracking and mobility prediction, a target vehicle can be passively and clandestinely tracked fairly efficiently. The contents of this chapter was published in [4].*

### 5.1 Introduction

With various applications directly targeting driver safety as well as other infotainment applications, the rapidly emerging vehicular networks are now envisioned to be as widely deployed in practice in the near future as other more mature wireless technologies. Although the main motive behind the emergence of vehicular networks was purely for driver/passenger safety, these networks can also be used for other purposes. One way in which vehicular networks can possibly be used is to assist crime investiga-

tion and prevention by providing law enforcement a new means to track suspects and criminals. Vehicular tracking algorithms naturally experience inherently different implications than those encountered at the conventional mobile networks. For instance, the nodes in the vehicular networks move more rapidly than the nodes in other mobile networks. Additionally, vehicular nodes are characterised by a somewhat limited motion freedom because their movements are usually constrained by the roadways structures and traffic regulation. This potentially introduces additional challenges for vehicular tracking algorithms requiring such algorithms to adapt to the peculiarity of vehicular movements, but it also improves their mobility predictability since nodes commute over pre-defined paths.

**Related Work.** In [18], Boukerche *et al.* discussed a few localisation techniques that can be used in vehicular networks along with their practical implications. The authors showed that most of the current localisation techniques suffer from inherent inaccuracies that may not be acceptable for some vehicular-based applications, especially those requiring precise location information. It is argued that in such situations, the most likely solution would be through data fusion where results from several localisation techniques are fused to improve the accuracy. Although most of the localisation techniques suitable for vehicular networks are GPS-based, not all vehicles are equipped with GPS receivers, but even if they do, such techniques become useless if GPS signals are not available (e.g., inside tunnels). Benslimane [16] addressed such situations and proposed an extension to the ODAM messaging dissemination protocol such that even vehicles that are not GPS-capable (temporarily or permanently) are still localised. One of the popular tracking techniques, especially suitable for vehicular scenarios, is map-matching, which matches the vehicle’s actual location (raw) data to maps. Barakat-soulas *et al.* [14] presented several such algorithms to exploit the vehicular trajectory information. However, this technique is not suitable for applications requiring strictly real-time tracking, such as law enforcement online tracking, which is the subject of this chapter. Other tracking techniques involve the installation of tracking tags on target vehicles [91], but such solutions, obviously, are not ad hoc and require preparation, which introduces additional implications and economical concerns.

**Chapter Outline.** This chapter is organised as follows. In section 5.2, we provide a brief background about vehicular networks, followed by our proposed vehicular tracking system, where we discuss localisation in section 5.3, motion prediction in section 5.3 and tracking in section 5.6. Finally, simulation results are presented in section 5.6.

## 5.2 Vehicular Networks

Generally, vehicular networks are based on ad hoc infrastructure and thus are usually called Vehicular Ad hoc Networks (VANET), which is considered a class of the more general Mobile Ad Hoc Networks (MANET). In the USA, the Federal Communications Commission (FCC) allocated 75 MHz spectrum in the 5.9 GHz band (5.850 GHz to 5.925 GHz) for vehicular communication which can be either Vehicle to Vehicle (V2V) or Vehicle to Infrastructure (V2I) communication<sup>1</sup>. V2V is also called Inter-Vehicle Communication (IVC) and is solely based on an ad hoc infrastructure where vehicles directly exchange information such as accident and congestion warnings. V2I, on the other hand, assumes the presence of pre-installed roadside components that vehicles can communicate with to retrieve information. However, installing such components is usually expensive, which limits their utility compared to V2V/IVC.

**Vehicular Communication.** Communication in vehicular networks can be based on several technologies, such as IEEE 802.16 (WiMAX), Bluetooth, IRA, ZigBee etc. However, the most popular of these are IEEE 802.11 (WiFi), or 3G via CDMA (Code Division Multiple Access) technology. While the former approach is simpler and cheaper to deploy, it suffers from reliability issues because WiFi was not originally designed to operate in environments with rapid movements. In contrast, 3G is a more robust solution for vehicular communication, but is also more expensive and difficult to deploy due to its centralised architecture; see [89] for a discussion about these approaches and the various proposed solutions.

In this chapter we only adopt a high-level generic algorithmic approach, we do not consider the peculiarities of the low-level details of vehicular communication.

**Vehicular Mobility.** Vehicular mobility models were discussed in section 3.4.2 where we showed how such models are being developed. However, instead of developing a model from scratch, we will adopt an existing vehicular mobility model from the literature, namely the IDM (Intelligent Driver Motion) model, which is a microscopic model [125]. We will consider IDM's two extensions, IDM-IM (IDM with Intersection Management) which extends IDM to handle the vehicles behaviour at intersections, and IDM-LC (IDM with Lane Changes) which extends IDM-IM allowing vehicles to change lanes and overtake.

---

<sup>1</sup>Another form of vehicular communication is the so-called Vehicle to Pedestrian/Person (V2P) where pedestrians are equipped with transponders to alert the drivers of nearby vehicles and so prevent pedestrian accidents. This form of vehicular communication is less relevant to our tracking purpose and will not be considered/discussed further.

### 5.3 Vehicular Localisation

The conventional localisation techniques, usually used in cellular and sensor networks as discussed in chapter 2, can also be used in vehicular networks (but with slightly degraded accuracy). Since we adopt a passive tracking approach, only RSS-based localisation is relevant because, unlike TOA and TDOA, it does not require a fully synchronised network and/or an active communication with the target. When adopting RSS, the strength of the signals emitted by the target vehicle is measured to estimate how far the target is. Even though RSS measurements are usually nonlinear, in our algorithm we adopt a free space radio propagation model, which assumes a direct line of sight path between vehicles. Such assumption does not severely degrade the accuracy of the localisation process because it is most likely that the only obstacles in our scenario will be the moving vehicles whose effects can be eliminated by averaging the measurements—the vehicular acceleration also affect RSS measurements, but we assume that this effect can be mitigated by adding random noise to the calculation.

Localising the target vehicle,  $C_s$ , is conducted by first recruiting 3 of its neighbouring vehicles. However, only two of these neighbours (tracking agents) will later track the target, while the other will be retired immediately once the localisation is completed. For the rest of the chapter, we will denote the tracking agents as  $C_a$  and  $C_b$  (such that  $C_b$  is a backup for  $C_a$  who is the main tracking agent) and the additional localising agent as  $C_l$ . We will also denote the tracker, who initiates the whole tracking process, by  $C_t$  (note that there may be more than one tracker). We assume that all agents are capable of localising themselves using, e.g., GPS, which is fairly common in today’s vehicles, or otherwise have access some location service allowing them to conveniently pinpoint their locations at any time. Localisation then proceeds in four steps (algorithm 5.1 illustrates this process):

- *Step 1: Recruitment.* The tracking agents  $C_a, C_b$ , as well as the localising agent  $C_l$  are recruited randomly by the tracker  $C_t$  who requests the RSS readings from nodes around  $C_s$  and recruits the nodes reporting closest to  $C_s$ ; see section 5.5 for more details about the initial and subsequent recruitments.
- *Step 2: Distance Measurements.* The distances between the recruited agents and the target are estimated by, e.g., the Friis equation [43], see section 4.3.1.
- *Step 3: Trilateration.* The location of the target is estimated through a geometric transformation that finds the intersection point of the circles formed by the 3 recruited agents with centres of the agents’ locations and radii of the respective distances between the agents and the target as measured in step 2.

- *Step 4: Retirement.* The localising agent  $C_l$  is retired.

---

**Algorithm 5.1** Vehicular Localisation
 

---

```

1: SET  $C_t \leftarrow$  Tracker {identify the tracker}
2: SET  $C_s \leftarrow$  Target {identify the target}
3: recruit( $C_a, C_b$ ) {recruit the tracking agent and its backup}
4: repeat
5:   recruit( $C_l$ ) {recruit the localisation agent}
6:   assign( $C_a \rightarrow C_1, C_b \rightarrow C_2, C_l \rightarrow C_3$ ) {for convenience, assign aliases}
7:   for  $i = 1$  to  $i \leq 3$  do
8:     measureDistance( $C_i, C_s$ ) =  $d_{i,s}$  {the distance between  $C_i$  and  $C_s$ }
9:   end for
10:  assign( $C_a \leftarrow C_1, C_b \leftarrow C_2, C_l \leftarrow C_3$ )
11:  localise( $C_s, d_{1,s}, d_{2,s}, d_{3,s}, C_a, C_b, C_l$ ) {localise  $C_s$  given the locations of  $C_a, C_b, C_l$ 
    and the distances between them and  $C_s$ }
12:  sendLocationUpdate( $C_t$ ) {send location updates to the tracker}
13:  retire( $C_l$ ) {retire the localisation agent}
14: until Tracking Expires
  
```

---

The localisation algorithm can also be used in conjunction with the prediction algorithm (section 5.4) to predict the next recruitment, assuming that the agent is able to measure/estimate its speed as described in section 5.4.1. The speed can also be estimated if we assumed that the agents are equipped with GPS or have access to some of location service where the average speed can be trivially estimated, or it can even be readily provided by any modern GPS software.

**Beaconing.** Some vehicular networks applications require the vehicles to transmit real time information such as their current position, speed etc. This information can be broadcasted by the vehicles in what is known as “beacons” [86]; any vehicle within the vicinity of the transmitting vehicle can then receive these beacons. Intercepting these beacons gives an even more convenient and straightforward way to find the location of the target, in which case there will no need to recruit more than one agent to track the target as no trilateration is required.

## 5.4 Mobility Prediction

A particularly important aspect to consider when tracking vehicles is their mobility behaviour because a good understanding of a particular mobility pattern will help predicting it. Generally, mobility prediction is slightly simpler in VANETs than MANETs since vehicular mobility is somewhat restricted to roadways and their movement pace is

usually limited to the maximum allowable speed of the corresponding roadway. Mobile nodes in MANETs, on the other hand, often perform unpredictable and unconstrained motion (though nodes are restricted by the terrain characteristics and their physical limitations). Moreover, vehicular motion clearly has a higher acceleration rate. In this section, we develop a vehicular mobility prediction algorithm to probabilistically predict the near-future movement of the target vehicle based on its current location and estimated speed, assuming that the target has already been localised. Our algorithm contributes two predictors:

- *Time predictor*: to estimate the elapsed time before a vehicle reaches the next intersection, and
- *Direction predictor*: to predict the direction the vehicle would most likely take past the next intersection.

As we will show later, predicting these two parameters allows the tracking agents to estimate how long they will be able to track the target for and then tailor the tracking process accordingly (e.g., by recruiting/retiring agents etc.).

#### 5.4.1 Time Prediction

To predict how long it will take a vehicle to reach the next intersection, the speed of that vehicle is estimated as illustrated in figure 5.1. In this figure,  $C_a$  estimates the speed of  $C_s$  by making two RSS measurements at times  $t_1$  and  $t_2$ , assuming, for simplicity, that the distance between the horizontally aligned vehicles in a roadway is  $I$ , which is easily obtainable. However, it is very unlikely that both vehicles  $C_a$  and  $C_s$  will be perfectly horizontally aligned, thus when taking the first RSS measurement  $RSS_1$ , we calculate the vertical distance ahead or behind  $C_a$  to be perfectly adjacent to  $C_s$  and then hypothetically adjust the position of  $C_a$  as if it is perfectly horizontally aligned with  $C_s$ , this is shown in figures 5.1(a) and 5.1(b):

$$N = \sqrt{(d_1(t_1))^2 - I^2} \quad (5.1)$$

Hence,  $C_a$  becomes  $C_{a+N}$  as shown in figure 5.1(b). Once  $C_a$  is appropriately adjusted, the second RSS measurement  $RSS_2$  takes place at time  $t_2$ . However, the distance obtained from  $RSS_2$  has to be adjusted with  $C_{a+N}$  by recalculating it as if it was measured by  $C_{a+N}$  instead of  $C_a$  which accounts for the extra distance  $N$  obtained in equation 5.1. Figure 5.1(c) illustrates this process graphically. Distance  $M$  is then calculated by the following equation (most of the equations below are based on the



or lagging the other. In figure 5.1, we assumed (without loss of generality) that vehicle  $C_s$  is leading  $C_a$  and is faster, so it will continue to lead. However, our algorithm will also properly accommodate other scenarios where  $C_a$  is the leading vehicle because we are measuring the absolute velocity of  $C_s$  without considering its relation to the location and speed of  $C_a$ .

Since we assumed that localisation takes place *before* the prediction algorithm, the lane that the target occupies will be known. The discussion above considered the situation where the  $C_a$  and  $C_s$  are not at the same lane; nonetheless, having them at the same lane is equally likely. If this is the case,  $C_a$  makes the first RSS measurement  $RSS_1$  at  $t_1$  and later makes the second RSS measurement  $RSS_2$  at  $t_2$ , then the distance  $C_s$  would have moved between these two measurements is simply the difference in time between  $RSS_2$  and  $RSS_1$  divided by the estimated speed. Moreover, if they are not in the same lane, it does not matter which one is in which lane (and which one is the leading vehicle); in figure 5.1, it happens that  $C_a$  is in the left and  $C_s$  is in the right lane, but if it was the other way round, the algorithm obviously still works. As noted earlier, to simplify the algorithm, we assume that the leading entity will continue to lead during the period between the  $RSS_1$  and  $RSS_2$  measurements, which is a reasonable assumption since this period is usually short enough to preserve the leading status.

### 5.4.2 Direction Prediction

Predicting the direction that the vehicle is likely to take through the next intersection is slightly more complex than predicting its speed bearing a probabilistic distribution. Direction prediction similarly assumes that the target vehicle has been accurately localised to the level of distinguishing the lane it occupies, which was not strictly important previously (in speed prediction, what matters was to find whether  $C_a$  and  $C_s$  are at the same lane, but not which one occupies which lane). We assume that all drivers adhere to the following basic traffic rules<sup>2</sup> as illustrated in figure 5.2:

**Rule 1.** *For a vehicle to turn right at an intersection, it must be present at the right lane of the roadway leading to that intersection.*

**Rule 2.** *For a vehicle to turn left at an intersection, it must be present at the left lane of the roadway leading to that intersection.*

**Rule 3.** *For a vehicle to go straight ahead at an intersection, it can be present at either the right or the left lane of the roadway leading to that intersection.*

<sup>2</sup>We do not consider U turns since some intersections do not allow them.

**Rule 4.** A vehicle must position itself in the appropriate lane according to its intended direction based on rules 1, 2 and 3 around 300 meters before the next intersection or half way through the roadway leading to the next intersection, whichever comes last. No lane changing for the purpose of accelerating movement pace (e.g., overtaking) beyond this point is permitted.

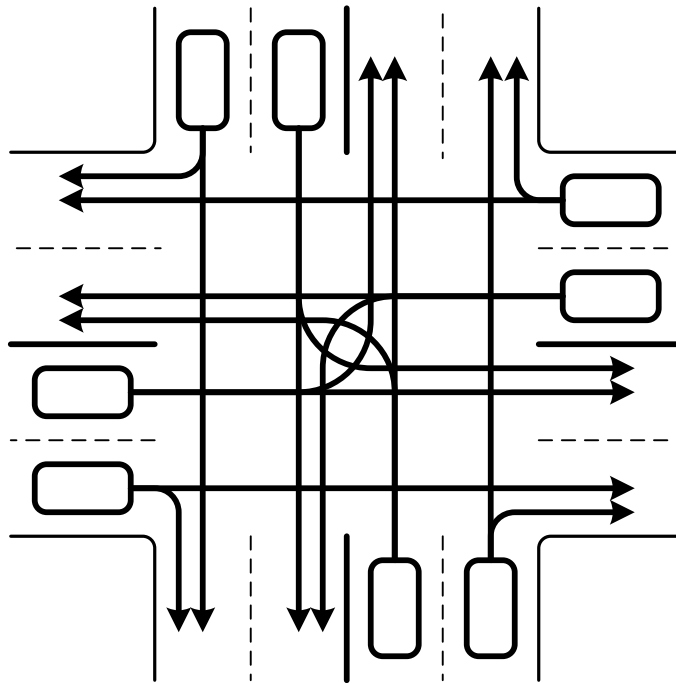


Figure 5.2: Intersection regulatory rules

These rules are applicable for 2-lane unidirectional roadways, which how our tracking scene is assumed to be (these are the most common city road infrastructure). It is realistic to assume that the target will follow these rules as the target (who is a suspect/criminal) will most likely not attempt to break the traffic rules so they do not raise suspicion. Assuming rules 1, 2, 3, 4 are fulfilled, vehicles are forced to start the lane changing process when reaching approximately  $n/4$  of the roadway of length  $n$ , or 400 meters away from the intersection, whichever comes last; this should allow sufficient time to complete the lane changing process without violating rule 4. We call the area between  $n/4$  and  $n/2$  (or similarly, the 100m between 400m and 300m to the next intersection) the *Critical Area* and apply the following predictions (note that the

probabilities are subjectively estimated based on common experience):

$$\text{Prediction} = \begin{cases} \Pr(C_i \rightarrow R) = 80\%, \Pr(C_i \rightarrow A) = 20\% & \text{if } C_i^L \xrightarrow{\text{CriticalArea}} C_i^R, \\ \Pr(C_i \rightarrow L) = 70\%, \Pr(C_i \rightarrow A) = 30\% & \text{if } C_i^R \xrightarrow{\text{CriticalArea}} C_i^L. \end{cases}$$

These predictions state that if a particular vehicle  $C_i$  is currently occupying the left lane ( $C_i^L$ ), and shifted to the right lane ( $C_i^R$ ) as it passes through the critical area, it is most likely that this shift was *imposed* by Rule 1, which means that the vehicle is very likely going to turn right ( $R$ ) at the next intersection because

1. the probability of turning left ( $L$ ) from the right lane (where the vehicle has shifted) is 0, and
2. if the vehicle intended to go straight ahead ( $A$ ), it would most likely have stayed in its previous lane without taking the burden of changing the lane because it is permitted to go straight ahead from either lanes.

Similar argument applies for vehicles shifting from the right lane ( $C_i^R$ ) to the left lane ( $C_i^L$ ) as they are passing through the critical area; however, differently in this situation, it is possible that the vehicle in the right lane shifted to the left lane to increase the pace of the movement<sup>3</sup> while actually intending to go straight ahead, this decreases the probability of turning left for this particular course of action. If, however, the vehicle did not change lane through the critical area, it has a 50% probability of taking either of the two permitted directions depending on its current lane while having a 0% probability of taking the banned direction (left for the right lane and right for the left lane). We also note that the algorithm can possibly be used to probabilistically guess from which direction the target came from by observing the lane it arrived at while leaving the intersection and entering a new roadway as shown in figure 5.2, but we do not discuss this *backward* mobility prediction here.

An interesting extension to this predictor (which we currently do not implement in our algorithm) is the integration of *Points of Interest* (POI) [36] where these are locations that are frequently visited by people; examples of POI are grocery stores, banks, restaurants, offices etc. Such locations can significantly influence the probability distribution of vehicular mobility. For example, we know that the probability of a vehicle in the right lane turning right is (currently) the same as the probability of going straight ahead if it had not shifted to the right lane as it passes through the critical area. However, if we further learnt that a particular POI is located at the right

<sup>3</sup>We assume that the left lane is the passing lane (also called fast lane). However, note that while in most countries, the left lane is the passing lane, in the UK, Australia, Japan and several other countries, the right lane is the passing lane.

direction, such information can possibly increase the probability of the vehicle turning right than going straight ahead. Moreover, the time of the day may greatly influence the effect of POI on the prediction probabilities; for example, offices will most likely be a popular POI only during daytime.

## 5.5 Vehicular Tracking

In forensic and law enforcement applications, tracking of vehicles is often required to be passive. This passivity requirement potentially eliminates the use of the active tracking and location-based systems that modern vehicles are usually equipped with, such as vehicle telematics. In our tracking scenario, a group of  $n$  privately connected trackers,  $C_{t(i)}$ , where  $i = 1, 2, \dots, n$  (e.g., police patrols), recruit a main tracking agent vehicle  $C_a$  and a backup tracking agent vehicle  $C_b$  from the public, to track a target vehicle  $C_s$ . We assume that, initially, at least one of the trackers  $C_{t(i)}$  is located at  $C_s$  range which allows it to recruit, possibly by visual estimation, suitable  $C_a$  and  $C_b$ . Subsequent recruitments are carried out by the active dissemination of a tracking software through vehicular networks in a manner similar to that discussed in chapter 4. At this initial stage, the recruiting  $C_{t(i)}$  also recruits an additional agent, the localisation agent  $C_l$ , which, along with the tracking agents, can localise the target as discussed in section 5.3 and illustrated in algorithm 5.1. The recruiting  $C_{t(i)}$  will also supply the addresses of the other valid  $C_{t(i)}$  to all recruited agents.

Once localisation of the target is completed, the localising agent is retired leaving  $C_a$  to be responsible for the rest of the tracking and backed up by  $C_b$ , whose task is to maintain  $C_a$  and takes over the tracking process should  $C_a$  suddenly fail.  $C_a$  will also be responsible for sending the location updates of  $C_s$  to one of  $C_{t(i)}$  whenever localisation is triggered.  $C_a$  must assure that its backup agent  $C_b$  is in both its and the target's range at all times by regularly monitoring  $C_b$ 's RSS and probing it for the target's RSS (this can be done by exchanging *alive* messages, see section 4.6.1) or alternatively it can request a neighbour list from  $C_b$  and check that both  $C_a$  and  $C_s$  exist, otherwise it has to recruit another  $C_b$ . At the beginning of the tracking,  $C_{t(i)}$  creates a tracking table,  $\mathcal{T}$ , to store records of the target's movements as received from  $C_a$ . This table is synchronised with all other  $C_{t(i)}$  as soon as an update is received at any  $C_{t(i)}$ . When a location update is available,  $C_a$  searches its range for any of the  $C_{t(i)}$  to update it; if multiple  $C_{t(i)}$  are found,  $C_a$  randomly chooses one<sup>4</sup>. Localisation is triggered in three cases (details follow):

<sup>4</sup>This potentially minimises traffic analysis attack by an attacker where the attacker detects the tracking process by observing the traffic between the agents and the trackers.

- when the alive period approaching expiration, or
- when the tracking agent start receiving weak signals from the target, or
- whenever one of the tracker forcefully requests a localisation update.

After running the prediction algorithms (section 5.4),  $C_a$  will have an estimate of how long it will be able to track the target, this period is called the *alive period*, and schedules the next localisation for near the expiration of that period. During the alive period,  $C_a$  will keep observing the RSS measurements from the  $C_s$ . If a specific lower threshold of RSS is reached, or the predicted alive period is near expiration,  $C_a$  will initiate a *probe* process, which involves sending requests to the neighbouring vehicles, supplying the address of  $C_s$  and asking for their RSS measurements, if they are able to receive emissions from  $C_s$ , then the vehicle with strongest RSS is recruited as a localisation agent  $C_l$ . Furthermore, any  $C_{t(i)}$  may at any time request  $C_a$  to localise  $C_s$  in which case  $C_a$  recruits  $C_l$  and forcefully executes the probe process.

If a location update is available but no  $C_{t(i)}$  is found in range,  $C_a$  creates a temporary tracking table,  $\mathcal{T}_{temp}$  and accumulates it while regularly probing for a valid  $C_{t(i)}$ ; once found,  $C_a$  transfers its  $\mathcal{T}_{temp}$  to that  $C_{t(i)}$  which, in turn, merges it with its (outdated) copy of  $\mathcal{T}$  and synchronises it with the other  $C_{t(i)}$ . Algorithm 5.3 illustrates this temporary tracking process. If  $C_a$  had to maintain a  $\mathcal{T}_{temp}$ , but then was triggered to recruit another  $C_a$  (as detailed above) and there is still no  $C_{t(i)}$  in range, the old  $C_a$  recruits a new  $C_a$  and hands off the tracking process to it along with the  $\mathcal{T}_{temp}$  which will be accumulated by the new  $C_a$  until a valid  $C_{t(i)}$  is found; the old  $C_a$  then retires itself. Algorithm 5.2 illustrates the basics of the tracking algorithm.

**Alternative Tracking.** Indeed, vehicular networks are rapidly emerging, but it is clear that not all on-road vehicles today support them. However, it is more common for vehicles to possess some sort of emission source(s) that can hence be tracked. These emission sources may be Bluetooth emissions from the vehicle itself (usually providing a handsfree calling service which can be found in most low-cost recently manufactured cars), or any other sort of short-range radiation from a device attached to the driver of the car, such as a mobile phone. This potentially allows us to use the conventional MANET tracking algorithms in a vehicular setting. However, due to the differences between VANET and MANET, the tracking algorithms of the latter need to be modified to accommodate the characteristics of the former. In particular, as discussed earlier, the pace in which vehicles move is significantly higher than nodes in the MANETs, which introduces a whole new class of complication. In this chapter, however, we only discuss tracking in vehicular networks assuming their presence and noting that alternative

**Algorithm 5.2** Online Vehicular Tracking

---

```

1: SET  $C_{t(i)} \leftarrow$  Trackers {identify the trackers}
2: SET  $C_s \leftarrow$  Target {identify the target}
3: SET minAP {threshold to define the start of expiration of AlivePeriod}
4: SET minRSS {define the min threshold values of the RSS}
5: Recruit( $C_a, C_b$ ) {recruit the tracking agent and its backup}
6: repeat
7:   AlivePeriod  $\leftarrow$  timePredictor( $C_s$ ) {find the alive period}
8:   if  $C_s.RSS(C_a) < \text{minRSS}$  or  $C_{t(i)}.localise = \text{true}$  or AlivePeriod  $< \text{minAP}$  then
9:     recruit( $C_l$ ) {recruit the localisation agent}
10:    localise( $C_s, C_a, C_b, C_l$ ) { $C_a, C_b, C_l$  Localise  $C_s$ }
11:    sendLocationUpdate( $C_{t(i)}$ ) {send location updates to one of the trackers}
12:     $\mathcal{T} \leftarrow$  updateTable(LocationUpdate)
13:     $C_{t(i)}.syncTable(\mathcal{T})$  {synchronise  $\mathcal{T}$  with all trackers}
14:    Retire( $C_l$ ) { $C_l$  is retired once localisation is completed}
15:   end if
16:   if  $C_a.RSS(C_s) < \text{minRSS}$  then {if  $C_s$  started reporting weak signals to  $C_a$ }
17:      $C_a.Probe$  {search for a new  $C_a$ }
18:     if  $\hat{C}_a.RSS(C_s) \geq \text{minRSS}$  and  $\hat{C}_a.RSS(C_b) \geq \text{minRSS}$  then
19:       recruit( $\hat{C}_a$ ) {recruit a new  $C_a$ }
20:     else if  $\hat{C}_a.RSS(C_s) \geq \text{minRSS}$  and  $\hat{C}_a.RSS(\hat{C}_b) \geq \text{minRSS}$  then
21:       recruit( $\hat{C}_a, \hat{C}_b$ ) {recruit new  $C_a$  and new  $C_b$ }
22:     end if
23:   end if
24:   exchangeAlive( $C_a, C_b$ ) {exchange alive messages between  $C_a$  and  $C_b$ }
25:   if  $C_a.RSS(C_b) < \text{minRSS}$  then {if  $C_a$  is losing its backup  $C_b$ }
26:     probe( $C_a$ ) {search for a new  $C_b$ }
27:     if  $C_a.RSS(\hat{C}_b) > \text{minRSS}$  and  $C_b.RSS(C_s) > \text{minRSS}$  then
28:       recruit( $\hat{C}_b$ ) {recruit new  $C_b$  when found}
29:     end if
30:   end if
31: until Tracking Expires

```

---

approaches *may* be pursued if vehicular networking support was not available, but we do not discuss such implications.

## 5.6 Simulation Results

To simulate a realistic vehicular network environment, we used the VanetMobiSim simulator [54] which generates vehicular mobility traces based on the IDM-LC model (see section 5.2), we then fed the traces to the NS-2 simulator [69] and run the tracking simulation. The localisation and tracking scenarios exhibit similarities to those sim-

**Algorithm 5.3** Temporary Tracking

---

```

1: SET  $C_{t(i)} \leftarrow$  Trackers {identify the trackers}
2: SET  $C_s \leftarrow$  Target {identify the target}
3: SET  $C_a \leftarrow$  Tracking Agent {identify the tracking agent}
4: while  $C_{t(i)} =$  unavailable do
5:    $\mathcal{T}_{temp} \leftarrow$  createTempTable( $C_a$ ) {create a temp table held by  $C_a$ }
6:   if locationUpdate = available then
7:      $\mathcal{T}_{temp} \leftarrow$  updateTable(locationUpdate)
8:   end if
9:   prob( $C_{t(i)}$ ) {search for trackers}
10:  if  $C_{t(i)} =$  available then
11:    sendTable( $C_{t(i)} \leftarrow \mathcal{T}_{temp}$ ) {send  $\mathcal{T}_{temp}$  to the tracker}
12:     $C_{t(i)}.syncTable(\mathcal{T}_{temp})$  {synchronise  $\mathcal{T}_{temp}$  with all trackers}
13:  end if
14: end while

```

---

ulated in chapter 4, so we only simulated the time prediction algorithm (simulating the direction prediction algorithm is difficult as it highly depends on behaviours that we cannot expect mobility traces generated by VanetMobiSim to preserve). In our simulation scenarios, we adopted a road-map that exhibits different roadway lengths, each with different speed limit. We further generated 4 mobility traces with different node densities to analyse the effect of node density on the accuracy of our time prediction algorithm. Randomly choosing a target vehicle, we predicted the time it would take the target to reach the next intersection, then compared this prediction with the time it actually later took that vehicle to reach that intersection as reported by NS-2. Furthermore, we run the time prediction algorithm for different lengths of time, ranging from 5 to 20 seconds, then observed how this affected the accuracy of the prediction. Figures 5.3,5.4,5.5,5.6 present the results in scenarios with node densities of 10, 30, 50, 100 nodes and running for 1000 simulation seconds (the x-axis represents number of trial, 10 trials in total, and the y-axis represents time). As shown in figures 5.3,5.4,5.5,5.6, there is a variation in performance at different scenarios. We summarise the main factors influencing the accuracy of the prediction as follows:

- *Node density.* As shown in figures 5.3 and 5.4, accuracy of the algorithm is moderately affected by node density. This is, in fact, what we would expect in real life scenarios; the more vehicles in a roadway, the harder to accurately predict their movements due to their irregular acceleration/deceleration behaviours.
- *Roadways length.* Roadways lengths also affect the accuracy of the algorithm because longer roadways allow for more acceleration fluctuation which may be

Simulation software	VanetMobiSim and NS-2
Simulation area	1000m <sup>2</sup>
Simulation duration	1000 seconds
Simulation runs	10
Node density	10–100
Node max. speed	varied
Mobility Model	IDM-LC
Communication technology	IEEE 802.11
Radio propagation model	Free space

Table 5.1: Simulation configurations

hard to model. Highways, for example, have fairly different characteristics than city roadways, both in terms of node density and traffic flow.

- *Speed limit.* Roadways with higher speed limits allow for an increased acceleration/deceleration fluctuation that may influence the accuracy of the algorithm.
- *Prediction Duration.* The time prediction algorithm measures the time the target takes when passing a specific distance, it will then add noise to the measurement to compensate for the future acceleration/deceleration before finally calculating the predicted time. However, the algorithm’s accuracy is improved when it observes the target for longer interval. This is illustrated in figures 5.3 and 5.4 where a prediction of 20 seconds interval (that is, the target’s movement is observed for 20 seconds before calculating the predicted time) usually yields better predictions than shorter intervals. Another way to improve the accuracy of the prediction is to average several prediction observations at different times.
- *Location of the prediction.* Despite the fact that the prediction process can take place anywhere through a roadway, in some cases, the exact point where the prediction process takes place may influence the accuracy of the prediction, especially for highways which will most likely have a higher speed limit than city roadways. For example, if the prediction algorithm was executed at the beginning of a highway, where the vehicles are starting to accelerate, it is likely that this acceleration is not representative for the rest of the journey.

Based on the above factors, it is clear that the tracking environment has an unavoidable influence on the tracking process. Careful modelling of the environment’s macroscopic features is, therefore, crucial to improve the accuracy of tracking. Infor-

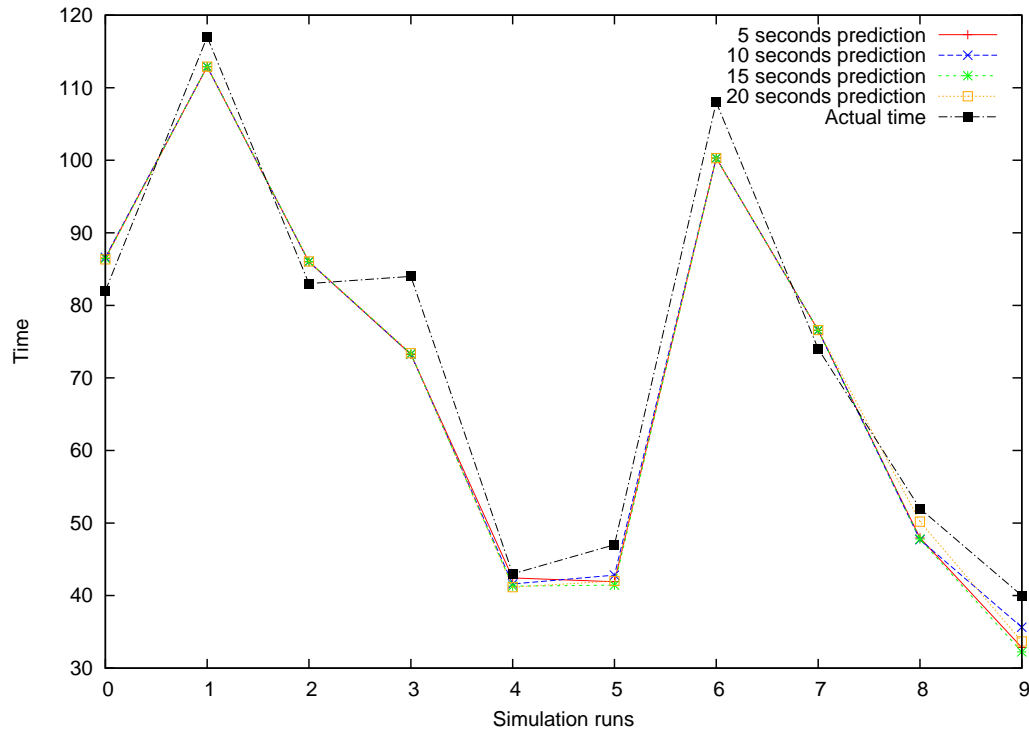


Figure 5.3: Simulation results for 10 nodes scenario

mation about the general structure and layout of roadways can easily be obtained from online databases, e.g., TIGER [126], while other road characteristics, such as node density, can be modelled (estimated) based on empirical observation of the concerned area or by subjective estimates based on the various characteristics of the area and previous experience, possibly also based on time of the day, as briefly discussed in section 5.4.2.

## 5.7 Vehicular Parameter Estimation

As discussed in section 2.5, all RF measurements made by tracking algorithms (especially the passive ones) include an unavoidable error margin. However, estimating the measurement parameters in the vehicular setting seem to be slightly more straightforward than estimating them in the pedestrian setting.

In vehicular tracking, it is not important to pinpoint the exact position of the vehicle since its mobility is usually restricted to the layout of the corresponding roadway. Compare with the pedestrian setting, where it is difficult to filter the estimations of an algorithm that has, e.g., 0.5 meters error margin because pedestrians have unrestricted mobility and can be anywhere within a vicinity of 0.5 meters. In contrast, estimations

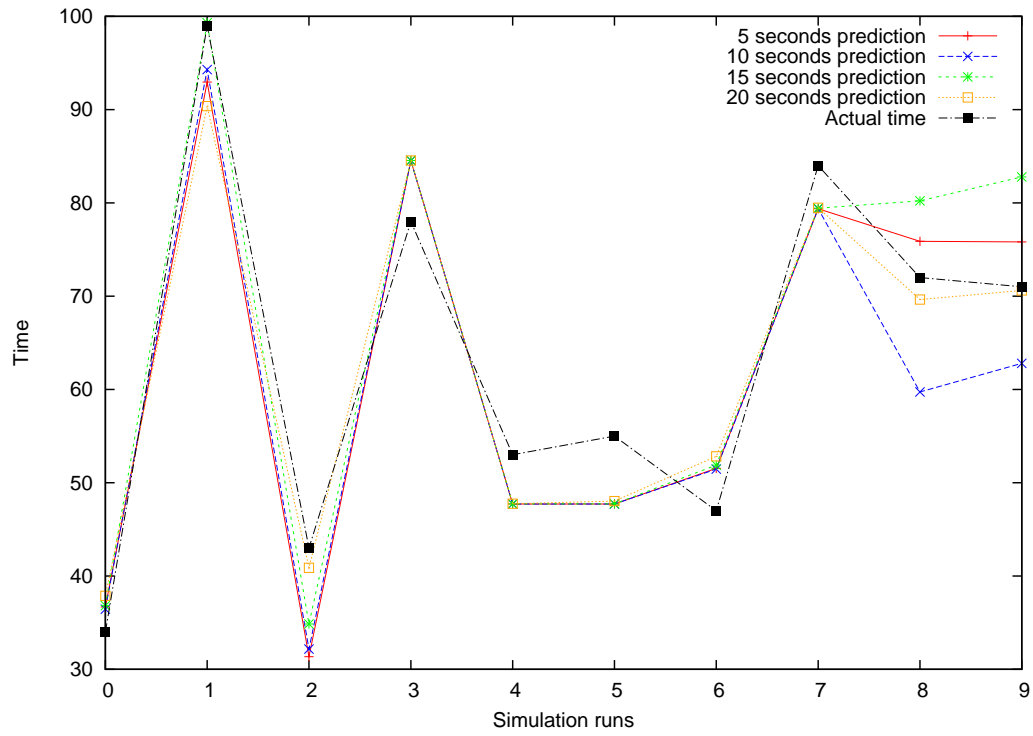


Figure 5.4: Simulation results for 30 nodes scenario

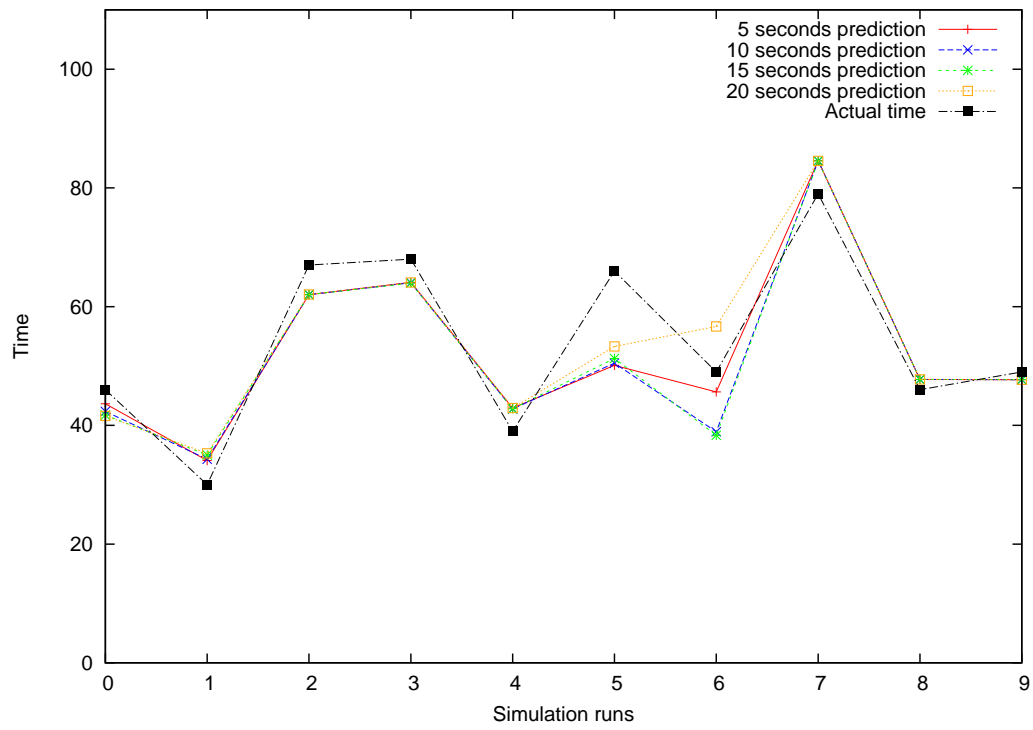


Figure 5.5: Simulation results for 50 nodes scenario

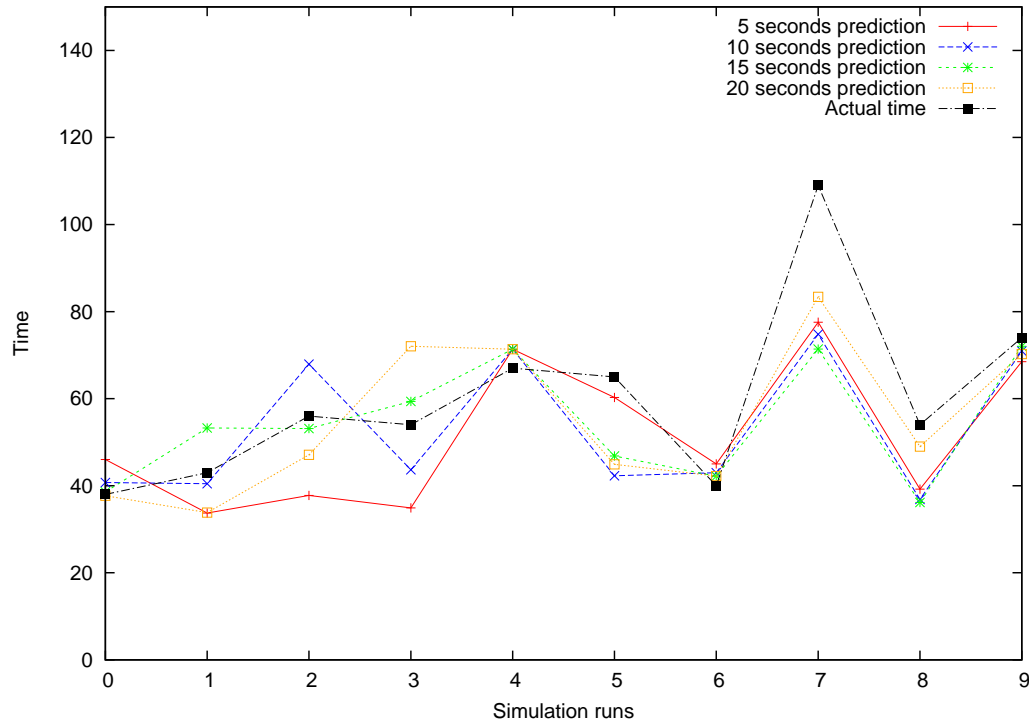


Figure 5.6: Simulation results for 100 nodes scenario

produced by the same algorithm in a vehicular setting, can be adjusted to roadways since it is unlikely that a vehicle will drive off-road. Inferences can then be made about the target’s mobility by observing the estimated time delay in each roadway (e.g., stopped to commit a crime). In chapters 6 and 7 we discuss such scenarios in an offline setting.

## 5.8 Summary

Vehicular networking is one of the most promising future technologies. Research in this area is becoming attractive due to the rapid emergence of vehicular applications. One emerging application that vehicular networks can be used for is tracking. Vehicular tracking is especially important for law enforcement to investigate and prevent crimes. In this chapter, we developed a set of online algorithms to passively localise, track and predict the near future movement of a target vehicle in real time. In particular, we adopt an agent-based tracking approach to track a target vehicle by systematically recruit agent vehicles via vehicular networks. We also proposed a motion prediction algorithm to predict the near future movements. This motion prediction algorithm can then be used by the tracking algorithm to adjust and maintain the tracking process.

## Part III

# Offline Forensic Tracking

## Chapter 6

# Bayesian Offline Vehicular Forensic Tracking

*In chapters 4 and 5 we discussed the online forensic tracking in pedestrian and vehicular settings. We showed how tracking can be conducted in these scenarios using an agent-based approach, which, as discussed in section 3.2.4, bears some privacy implications. In this rest of the thesis we consider the more common and less privacy-controversial scenario of tracking a target in an offline manner. In this chapter, we consider a vehicular tracking scenario and propose an offline post hoc vehicular trace reconstruction algorithm that can accurately reconstruct vehicular mobility traces of a target entity by fusing the corresponding available visual and radio-frequency surveillance data. The algorithms provide a probabilistic treatment to the problem of incomplete data by means of Bayesian inference. We realise that it is very likely that an extracted trace of a target entity would contain gaps (due to missing trace data), so we try to probabilistically fill these gaps. This allows law enforcement agents to conduct offline tracking while characterising the quality of available evidence. The contents of this chapter was published in [6].*

### 6.1 Introduction

Wireless mobile devices such as mobile phones and laptops can provide useful geolocation information. However, while such data allows for tracking mobile entities, collecting it from multiple sources and logically correlating them often creates new challenges for digital forensics practitioners. Generally, tracking can either be online

or offline. Online tracking involves observing the movement of a target entity in real time and is usually reactive (and adaptive) according to the target’s behaviours; online tracking has been extensively discussed in chapters 4 for pedestrian setting, and in chapter 5 for vehicular setting. However, real-time (online) tracking is often difficult and requires considerable amount of preparation, especially in criminal scenarios where such process needs to take place clandestinely (in some cases it is even not possible due to the unpredictable nature of crimes). Offline tracking, on the other hand, entails the extraction of the target’s movement traces from raw tracking data and analysing them appropriately. In most cases, offline tracking encounters a problematic phenomenon that we call *tracking gaps*. These gaps represent missing tracking data spanning particular areas over specific periods of time where tracking of the target was not possible, usually due to constrains of tracking resources. In this case, it is important to try filling these gaps by probabilistically selecting the routes the target would have most likely taken between the end points of the gaps; these end points are called the *Ingress*, which is the point at which tracking of the target was lost, marking the beginning of a gap, and the *Egress*, which is where tracking later resumed, marking the end of that gap. While such missing data scenarios are certainly important to consider for tracking individuals, we instead discuss vehicular tracking as this is the most common means of transportation; we leave the offline tracking of pedestrians as an extension to this thesis, see chapter 8. Vehicular traces can be collected by explicitly tracking the target vehicle, or by extracting data from the existing traffic infrastructure (e.g., CCTV) that was not originally installed for tracking. Either way, we assume that this data was collected passively since this is required by most law enforcement applications. Beside exhibiting tracking gaps, such data will certainly contain different types of measurement errors, thus we develop a probabilistic Bayesian-based method to reconstruct the target trace.

**Related Work.** Incomplete (or missing) data is a common problem in many contemporary applications. Expectation-Maximization and Data Augmentation methods (along with their improved variants) are among the most popular statistical treatments for the missing data problem [122]. Similarly, scene reconstruction (which usually deals with missing data) has been a highly active area of research over the past few decades, especially for forensics and crime investigation purposes. In general, most of the research in this area involves reconstructing scenes from images. For example, in [19] Calbi *et al.* proposed a set of computer-vision-based algorithms that are able to reconstruct a 3D scene of multiple moving objects. This system utilises a set of pre-installed cameras surveilling the area to be reconstructed, and thus can only be used at these

heavily monitored areas which degrades its flexibility. In [48], Greenhill *et al.*, tried to overcome the inflexibility of the fixed camera surveillance systems by proposing algorithms that are able to reconstruct scenes by collecting observation streams from mobile cameras mounted on buses. Moreover, in [27], Conaire *et al.* discussed how to fuse image-based and RF-based (radio frequency) localisation to improve the overall accuracy of the process. The authors tested their mechanism in a museum environment where visitors are equipped with devices containing a portable camera. The device automatically captures some photos of its surroundings and compares them with a database of images to identify its current location. This information is then fused with an estimation of signal strength from several wireless access points distributed around the museum, where compared to signal strength histograms that were created for various locations in the museum and stored in a database. Although most of these algorithms were proposed to track individuals, we are explicitly concerned with vehicular tracking in this chapter. As mentioned in 5.1, Brakatsoulas *et al.* discussed the feasibility of reconstructing the movement patterns of objects by observing their GPS tracking information [14]. Their algorithm adopts the so-called *map matching* technique where the tracking information of the objects (which are vehicles in this case) are analysed and matched to a road-map. Nevertheless, these algorithms usually suffer from various error sources [73], which motivated the development of improved techniques such as map-matching based on Artificial Neural Network [130]. This chapter adapts a somewhat similar approach to map-matching but for the purpose of reconstructing a full tracking trace of a particular entity, not quite concerned about accurate localisation since it suffices to know that a target vehicle has been in a particular roadway to draw conclusions about how its trace can be reconstructed.

**Chapter Outline.** This chapter is organised as follows. In section 6.2 we describe how to prepare a trace for the reconstruction process. This preparatory phase uses algorithms to fuse traces from different sources (RF-based and visual-based) and so minimising the number of tracking gaps in a target’s trace. Our trace reconstruction algorithm is proposed in section 6.3 in two phases, phase 1 in section 6.3.1 and phase 2 in section 6.3.2, followed by a discussion about its accuracy in section 6.5. Finally, in section 6.4 a few simulation results are presented and discussed.

## 6.2 Trace Fusion

Vehicular traces are records containing motion information of vehicles over a particular area and during a specific period of time. These traces can be collected by various tools,

such as cellular tracking, Radar etc. beside the passive techniques presented in chapter 5. However, data from other tools like CCTV (Closed Circuit Television), which are not originally built for tracking, can be used to improve the existing tracking traces. We will assume that we have two sets of tracking data belonging to a particular target:

- RF tracks: collected by RF-based tracking techniques<sup>1</sup>, e.g., chapter 5,
- visual tracks: retrieved from CCTV cameras located at the tracking scene.

These two sources (RF and visual tracks) are natural passive tracking sources. Tracks from active sources (where the target was actively tracked) can be incorporated too if available. Regardless of the source of tracks, there are only two possible tracks that can be obtained, either a range (distance) over which the target was observed, or a single point at which the presence of the target was detected (multiple single points may form a range). In our context, we call the first set of tracks RF tracks, assuming that they were most likely generated by passive (or active) RF-based tracking mechanisms, while calling the other set of tracks, visual tracks, assuming that they were generated by image/video-based tools, such as CCTV cameras. However, note that range tracks do not necessarily have to be generated by RF-based mechanisms, and as discussed above, indeed in some cases advanced computer vision algorithms can generate such tracks by analysing footage from CCTV cameras, for example. Similar argument holds for the visual tracks, where they can potentially be generated by passive/active RF-based mechanisms.

Before commencing the actual trace reconstruction process (i.e., the offline tracking), we first try to fuse any RF and visual tracks we may have. However, this fusion is just an auxiliary phase and is not necessarily required to proceed to the reconstruction process, that is, we assume that we have access to both RF and visual tracks, and the individual tracks exhibit tracking gaps, but when fused, we hope that the number of these gaps is minimised or at least they shrink.

Interpreting RF tracking data is straightforward, as a minimum, each record consists of time, vehicle ID and location. On the other hand, to interpret the visual tracks, we assume prior knowledge of the fixed locations of the CCTV cameras and that they have a somewhat narrow recording angular distance, then we can estimate the location of a detected target to be the location of the detecting camera. One possible detection method is to observe the vehicle's plate numbers, this can be done by viewing images from CCTV footage and compare them to images containing the target's plate number. Computer vision techniques such as SURF [15], perform similar image-based detection

---

<sup>1</sup>RF tracks can also be generated from multiple CCTV footage using some advanced computer vision algorithms that can estimate how fast and how long the target was moving across the camera vicinity.

by identifying interest points in the images. The main difference between RF and visual tracks, though, is that RF tracks represent continuous movements of the vehicle for a period of time (a set of chronological tracking records), while the visual tracks represent fixed locations of the target where it was detected.

In this fusion process, we further aim to construct a trace of the target containing gaps only between intersections. However, in practice, only partial tracks may be available (due to resource constraints), where the target was unobservable at some points through the road, potentially creating tracking gaps that are not bounded by intersections. In such cases, the available tracks of these roadways (with incomplete traces) may be: (a) visual only, (b) RF only, or (c) both RF and visual. In all these situations, we run a prediction algorithm to estimate the time it took the target to reach the next intersection and thereby filling the whole roadway; algorithm 6.1 provides a pseudocode for this fusion process and figure 6.1 illustrates it graphically.

Situation (a) provides very little information for this algorithm to work, so such roadways are ignored and marked as *semi-incomplete* which basically indicates that the target was observed in this roadway (this information may prove useful in the trace reconstruction algorithm; see section 6.3), as illustrated in figure 6.1(a). To simplify the discussion, in situations (b) and (c) we assume that there are only single RF and/or single visual tracks per roadway, but of course the discussion can be extended to consider multiple RF/visual tracks without introducing any change in the algorithm. The prediction algorithm is based on estimating the speed of the target over the period covered by the available tracking information, which will then be used to predict the time it would have taken the target to reach the next intersection given the remaining distance of the corresponding roadway.

In situation (b), we have a single RF track range, which we expand to fill the whole roadway (details below), as illustrated in figure 6.1(b). Similarly, in situation (c), if the visual track is within the RF track range, it provides no extra information (essentially becoming situation (b)) and the visual tracks can be ignored, but if the RF and visual tracks are apart, we expand the range of the RF to include the visual track; this is illustrated in figure 6.1(c). Usually, the longer the available tracking range, the better prediction we can expect. After connecting the RF and visual tracks, the result is a tracking range and the predicted time to the next intersection can be calculated as follows:

$$t_i = \frac{n - (d(R)_0 + d(R)_1)}{S} = \frac{(t(R)_1 - t(R)_0)(n - (d(R)_0 + d(R)_1))}{d(R)_1 - d(R)_0} \quad (6.1)$$

where  $S = \frac{d(R)_1 - d(R)_0}{t(R)_1 - t(R)_0}$  is the speed of the vehicle  $i$  (the target),  $d(R)_0$  and  $d(R)_1$  denote the beginning and the end of the tracking range, respectively, and  $t(R)_0$  and  $t(R)_1$  denote the times at which the tracking range started and ended, respectively. Equation 6.1 assumes  $d(R)_0 = d_{I,i}$  (where  $d_{I,i}$  is the beginning of a roadway), but if this is not the case, then this means that the tracking range did not start at the beginning of the roadway, so we need to do a *backward* prediction (connecting the beginning of the range with the beginning of the roadway) as well as a *forward* prediction (connecting the end of the range with the end of the roadway), both of which can easily be done in a similar manner. Furthermore, if  $d(R)_1 = d_{E,i}$  (where  $d_{E,i}$  is the end of the roadway), then only backward prediction is required. In any case, once we know the average speed of the target (which can be obtained from  $d(R)_0$ ,  $d(R)_1$ ,  $t(R)_0$ ,  $t(R)_1$ ), then given a distance, we can modify equation 6.1 to compute the time it would take the target to drive that distance, regardless of whether it is forward or backward.

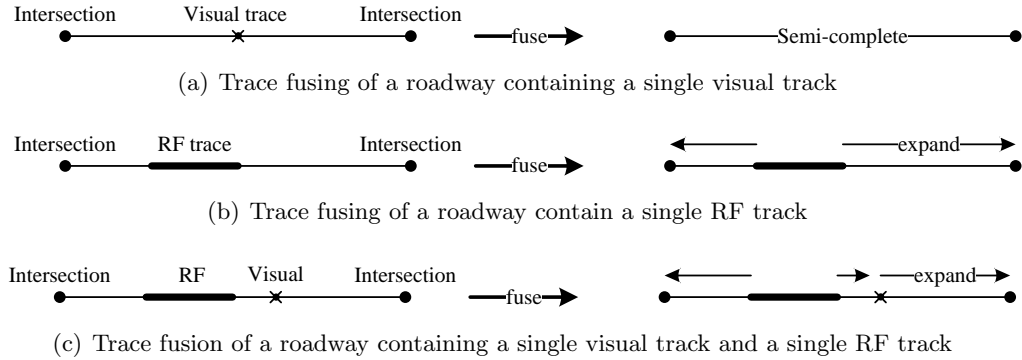


Figure 6.1: Trace fusion of RF and visual based tracks

## 6.3 Trace Reconstruction

Realistically, even after RF-visual fusion, the target's tracking trace will still most likely exhibit tracking gaps. In this section, we propose a 2-phase algorithm to probabilistically fill these gaps. The algorithms assume that the underlying layout of the tracking scene resembles a Manhattan-grid (see figure 6.2); in chapter 7 we generalise the algorithm to consider non-Manhattan-grid-like areas. In phase 1, the two end points (Ingress and Egress) of a gap along with all possible routes between them are identified. Then in phase 2, the driving behaviour of the target is analysed based on the target's available tracking traces. This will later be used by the reconstruction algorithm to probabilistically select the connecting routes that the target would most likely have taken through the gaps to finally obtain the full reconstructed trace.

**Algorithm 6.1** Trace Fusion

---

```

1:  $\mathcal{R} \leftarrow \text{obtain}(\text{Roadways})$  {obtain the set of all roadways of a particular area}
2: repeat
3:    $\text{chooseRandom}(R \leftarrow \mathcal{R})$  {choose a random roadway from  $\mathcal{R}$ }
4:    $\text{obtainTracks}(R)$  {obtain all available tracks of roadway  $R$ }
5:   if  $R.\text{visual}$  and not  $(R.\text{range})$  then {if  $R$  contains visual tracks but not RF}
6:      $\text{mark}(R) = \text{semi-complete}$ 
7:   else if  $R.\text{range}$  and not  $(R.\text{visual})$  then
8:      $\text{extendRange}(R)$  {call the Range Expansion procedure}
9:   else if  $R.\text{visual}$  and  $R.\text{range}$  then
10:     $\text{extend}(R.\text{range}, R.\text{visual})$  {extend the range to include the visual track}
11:     $\text{extendRange}(R)$  {call the Range Expansion procedure}
12:   end if
13:    $\text{remove}(R, \mathcal{R})$  {remove roadway  $R$  from  $\mathcal{R}$ }
14: until  $\mathcal{R} = \perp$  {repeat until all roadways in  $\mathcal{R}$  are exhausted}

```

---

**Procedure 6.2** Range Expansion Procedure

---

```

1:  $\text{extendRange}(R)$ 
2: if  $R.\text{range.begin} = R.\text{begin}$  then
3:    $\text{forwardExpansion}(R.\text{range})$  {expand the range forward to the end of  $R$ }
4: else if  $R.\text{range.end} = R.\text{end}$  then
5:    $\text{backwardExpansion}(R.\text{range})$  {expand the range backward to the beginning of  $R$ }
6: else if  $R.\text{range.begin} \neq R.\text{begin}$  and  $R.\text{range.end} \neq R.\text{end}$  then
7:    $\text{forwardExpansion}(R.\text{range})$ 
8:    $\text{backwardExpansion}(R.\text{range})$ 
9: end if

```

---

**6.3.1 Phase 1: Routes Identification**

In this phase, the end points,  $P_{I,G_i}$  and  $P_{E,G_i}$  (the Ingress and Egress, respectively), of a gap,  $G_i$  (where  $i = 1, 2, \dots, n$  for a trace with  $n$  gaps), along with the possible routes between  $P_{I,G_i}$  and  $P_{E,G_i}$ , are identified, assuming that  $P_{I,G_i}$  and  $P_{E,G_i}$  correspond to intersections. However, it may be computationally expensive (or even infeasible) to identify *all* the possible routes between  $P_{I,G_i}$  and  $P_{E,G_i}$  when having a large tracking area, regardless of the size of the gap. Thus, we restrict the area under consideration by setting boundaries around the tracking gap, this bounded area is called the *search area* which we expect to cover the most probable routes a target would probably have taken through the gap. Figure 6.2 graphically illustrates a sample tracking trace of a target with four (bounded) tracking gaps. Since we assumed that the tracking area is a Manhattan grid, the search area will be of a rectangular or square shape containing both  $P_{I,G_i}$  and  $P_{E,G_i}$ . Thus, we have two possible orientations of the positions of the



In conventional flooding algorithms, the goal is to deliver a message to all nodes within a particular area by configuring every node to forward every message it receives to all other nodes it is connected to except the node it received the message from. Similarly, BRC uses a message-passing broadcast mechanism to discover the routes between  $P_{I,G_i}$  and  $P_{E,G_i}$ . When the BRC algorithm is first executed at  $P_{I,G_i}$ , it generates as many messages as there are exit points attached to  $P_{I,G_i}$ , each message represents a separate flow. These flows then multiply at every intersection they reach as long as it is a search vertex (creating as many flows as there are exit points attached to that vertex). This process continues until all flows are terminated. A flow terminates when it reaches: (1) a non-search vertex, (2) the  $P_{I,G_i}$  or (3) the  $P_{E,G_i}$ . When a flow terminates it raises a special tag with a value of 1, if the flow reached  $P_{E,G_i}$ , 0 otherwise. If the termination value was 1, the corresponding flow sends a message back to  $P_{I,G_i}$  reporting its traversed path. Algorithm 6.3 illustrates the BRC algorithm, and figure 6.3 shows an example scenario of four routes found through a gap using BRC (note that here we assume a bi-directional roadways, or a roadway consisting of multiple lanes, where vehicles can flow both forward and backward).

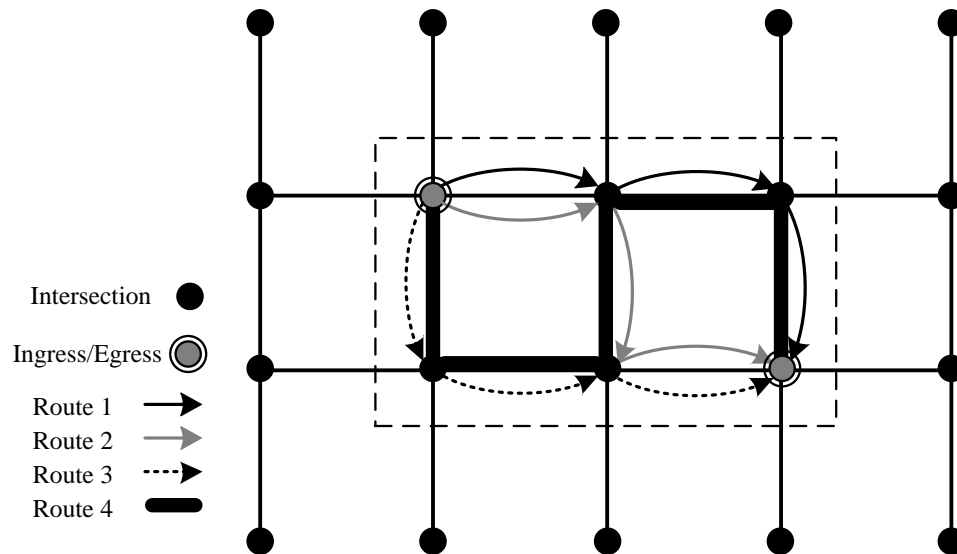


Figure 6.3: Possible routes through a sample gap

It is easy to see that the algorithm will both terminate and find all possible routes from  $P_{I,G_i}$  to  $P_{E,G_i}$  (that are within the search area). Initially, the algorithm is executed at  $P_{I,G_i}$  where it has four possible directions to send the flows through (in this case, the flow did not come from a particular node, so it is sent to the 4 possible direction off  $P_{I,G_i}$ ). At least one flow will hit a search vertex and will further propagate since at least one direction out the  $P_{I,G_i}$  leads to a search vertex. Also, since a flow cannot

terminate as long as it is propagating through search vertices and that all vertices will propagate a received flow out all their possible directions (except the one it came from), it is guaranteed that all search vertices will be visited and only flows that terminate at  $P_{E,G_i}$  will report back to  $P_{I,G_i}$ . The BRC algorithm is a basic bounding algorithm that assumes a manhattan grid, we will later generalise it in chapter 7.

---

**Algorithm 6.3** Bounded Route Counter (BRC)
 

---

```

1: SET  $P_{I,G_i} \leftarrow$  Ingress {identify Ingress}
2: SET  $P_{E,G_i} \leftarrow$  Egress {identify Egress}
3: SET Links {array to store the routes connecting  $P_{I,G_i}$  and  $P_{E,G_i}$ }
4: SET Flow {array containing the individual roadways of a route}
5: ExitPoints {number of exit points attached to a vertex, usually 4}
   {now call the markSearchVertices procedure to identify the search vertices}
6: markSearchVertices( $P_{I,G_i}, P_{E,G_i}$ )
7: flood( $P_{I,G_i}$ ) {initiate the flooding process at  $P_{I,G_i}$ }
8:  $x = P_{I,G_i}$  {store the flood initiation point in  $x$ }
9: label Loop
10: for  $i = 1$  to  $i = \text{exitPoints} - 1$  do
11:   if  $i.\text{nonSearchVertex} = \text{True}$  or  $i.\text{Ingress} = \text{True}$  then
12:      $i.\text{tag} = 0$  {set tag to 0 if the Ingress or non search vertex is reached}
13:   else if  $i.\text{searchVertex} = \text{True}$  then
14:     Flow = append(Edge( $x, i$ )) {update the Flow variable}
15:     flood( $i$ ) {create new flows and send them out the current vertex}
16:      $x = i$  {update  $x$  which contains the flood initiation point}
17:     Goto Loop
18:   else if  $i.\text{Egress} = \text{True}$  then
19:      $i.\text{tag} = 1$  {set tag to 1 if the Egress is reached}
20:     addRoute(Links, Flow) {add the current route to the Links array}
21:     purge(Flow) {reset the Flow variable}
22:   end if
23: end for

```

---



---

**Procedure 6.4** Search Vertices Marking Procedure
 

---

```

1: markSearchVertices( $P_{I,G_i}, P_{E,G_i}$ )
2: if aligned( $P_{I,G_i}, P_{E,G_i}$ ) = True then
3:   return  $P_{I,G_i} - 1, P_{I,G_i} + 1, P_{E,G_i} + 1, P_{E,G_i} - 1$ 
4: else if diagonal( $P_{I,G_i}, P_{E,G_i}$ ) = True then
5:   if leftDiagnoal( $P_{I,G_i}, P_{E,G_i}$ ) = True then {if  $P_{I,G_i}$  is at the left of  $P_{E,G_i}$ }
6:     return  $P_{I,G_i}, P_{I,G_i} + 1, P_{E,G_i}, P_{E,G_i} - 1$ 
7:   else if rightDiagnoal( $P_{I,G_i}, P_{E,G_i}$ ) = True then {if  $P_{I,G_i}$  is at the right of  $P_{E,G_i}$ }
8:     return  $P_{I,G_i} - 1, P_{I,G_i}, P_{E,G_i} + 1, P_{E,G_i}$ 
9:   end if
10: end if

```

---

In the current scenario we expect that the size of the search areas is minimised due to the visual-RF fusion process, and so adopting more sophisticated algorithms, such as branch-and-bound, will probably just slightly enhanced the efficiency of the whole route identification process at the cost of unnecessary overall complication. However, we will adopt a branch-and-bound like approach while developing the more general trace reconstruction algorithm for multi-modal setting in chapter 7.

### 6.3.2 Phase 2: Routes Analysis and Selection

At this stage, all routes between  $P_{I,G_i}$  and  $P_{E,G_i}$  are identified; we will denote the routes by  $R_j^i$ , where  $i = 1, 2, \dots, n$  and  $j = 1, 2, \dots, f(G_i)$ , for the  $j^{\text{th}}$  route of the  $i^{\text{th}}$  gap. Also, each  $R_j^i$  consists of  $f(R_j^i)$  roadways, where  $f(R_j^i) = 1, 2, \dots$ . The function  $f$  can be thought of as an *overloaded* function which behaves differently depending on its input (arguments), that is, given a gap  $G_i$ ,  $f$  returns the number of routes through  $G_i$ , but given a route  $R_j^i$ ,  $f$  returns the number of roadways forming  $R_j^i$ .

$$f(X) = \begin{cases} \text{routes in } X & \text{if } X \text{ is a gap} \\ \text{roadways in } X & \text{if } X \text{ is a route} \end{cases}$$

In this phase, we calculate the probability that a particular route<sup>2</sup> was taken by the target given the actual time that the target spent while traversing the corresponding gap, which is the time difference between  $P_{I,G_i}$  and  $P_{E,G_i}$  as obtained from the original incomplete traces. These analyses are based on a basic Bayesian inference where we first study the driving behaviour of the target and then assign probabilities for each possible route through the gap. By investigating the target's driving behaviour, we essentially try to model its mobility to identify movement patterns, which is then used in the route reconstruction algorithm. However, before executing the route reconstruction algorithm, we first check whether any of the routes contains *semi-incomplete* roadways. Recall from section 6.2 that a semi-incomplete roadway is a roadway in which the target was observed at but could not be included in the RF-visual fusion process. If a particular semi-incomplete roadway falls between  $P_{I,G_i}$  and  $P_{E,G_i}$ , we can safely ignore any route not passing through that roadway because we know that the target was indeed in that semi-complete roadway; this will potentially minimise the number of routes under consideration. Once the possible routes are identified, the algorithm proceeds in 5 steps:

<sup>2</sup>We assume that targets will not make U turns at any intersection.

**Step 1: Mean and variance.** After identifying the routes and the roadways each route is composed of, we search the available trace of the target for roadways with similar lengths as those comprising the routes between  $P_{I,G_i}$  and  $P_{E,G_i}$ . Then, we calculate the mean (average) of the sum of times the target spent driving those roadways and assign the result to the corresponding roadways of the gap's routes. That is, if a route  $R_j^i$  through the gap  $G_i$  consists of  $f(R_j^i)$  roadways  $R_j^i = r_1, r_2, \dots, r_{f(R_j^i)}$ , we consider each roadway  $r$  individually and search the available trace of the target (the records in which the target was observed) for roadways with approximately similar lengths as those in  $R_j^i$  and extract the time periods it took the target to pass those roadways, we then take the average and assign it to the corresponding roadways in  $R_j^i$ . Preferably, we find more than one route in the available target trace corresponding (in terms of length) to each route in  $R_j^i$  so we can take their average, the more samples we are able to obtain, the better averaging accuracy we would expect. Additionally, it is also desirable that those extracted routes (from the available target trace) are located geographically close to those of  $R_j^i$  and with similar environmental characteristics (see section 6.5). Once this is done for  $R_j^i$ , we repeat the process for routes through  $G_i$ . In this step, we aim to find the mean (average) and variance of the target's driving time for each route through the gap. Formally, the mean of a route  $R_j^i$  consisting of  $f(R_j^i)$  roadways is:

$$\mu_{R_j^i} = \frac{1}{f(R_j^i)} \sum_{x=1}^{f(R_j^i)} \left( \frac{1}{S(x)} \sum_{y=1}^{S(x)} t_{E,y} - t_{I,y} \right) + \omega_x \quad (6.2)$$

where  $S(x)$  is the number of roadways from the available target's traces with the same length as roadway  $x$ , ( $x = 1, 2, \dots, f(R_j^i)$ ), and  $\omega$  is the delay factor which may be different for different roadway, see section 6.5 for a discussion about how to model this parameter. Since it is easy to extract information about roadways with available tracking information, it is possible to find when the target entered and exited each one of these roadways ( $t_{I,y}, t_{E,y}$ , respectively, for roadway  $y$ ). Next, we calculate the corresponding variance of every route as follows:

$$\sigma_{R_j^i}^2 = \frac{1}{f(R_j^i)} \sum_{x=1}^{f(R_j^i)} \left( \left( \frac{1}{S(x)} \sum_{y=1}^{S(x)} t_{E,y} - t_{I,y} \right) + \omega_x - \mu_{R_j^i} \right)^2 \quad (6.3)$$

We then use Bayes' theorem to calculate the probabilities that the target took each route between the Ingress and the Egress of a gap, and select the route with the highest probability as the connecting route. We base our probability calculations on the time difference between the Ingress and the Egress of the gap  $t_{G_i} = t_{E,G_i} - t_{I,G_i}$  as obtained

from the original trace<sup>3</sup>. That is, for every route, we calculate the probability that the target could have taken that route given the time we obtained by observing the records at  $P_{I,G_i}$  and  $P_{E,G_i}$  (the time the target spent in gap  $G_i$ ):

$$\begin{aligned}\Pr(\text{route}|\text{time}) &= \frac{\Pr(\text{time}|\text{route}) \cdot \Pr(\text{route})}{\Pr(\text{time})} \\ \Pr(R_j^i|t_{G_i}) &= \frac{\Pr(t_{G_i}|R_j^i) \cdot \Pr(R_j^i)}{\Pr(t_{G_i})}\end{aligned}\quad (6.4)$$

where  $\Pr(t_{G_i}|R_j^i)$  is the conditional probability that given the target took the route  $R_j^i$ , he spent  $t_{G_i}$  driving it, which is calculated for every route (with a fixed  $t_{G_i}$ ),  $\Pr(R_j^i)$  is the prior probability that a target will take the route  $R_j^i$ , and  $\Pr(t_{G_i})$  is the marginal probability. Steps 2 to 4 below show how these probabilities are calculated.

**Step 2: Conditional probability.** The probability that the target spent the time  $t_{G_i}$  while driving from  $P_{I,i}$  to  $P_{E,i}$  through gap  $G_i$  given that he took route  $R_j^i$  is:

$$\Pr(t_{G_i}|R_j^i) = \frac{1}{\sqrt{2\pi\sigma_{R_j^i}^2}} \exp\left[-\frac{(t_{G_i} - \mu_{R_j^i})^2}{2\sigma_{R_j^i}^2}\right]\quad (6.5)$$

This probability is calculated using Gaussian probability density function assuming that the underlying process is Gaussian [94], that is, the driving behaviour of the target will most likely follow a Gaussian distribution or can be estimated as Gaussian. In fact, it is easy to see that this process is Gaussian because typical driving behaviour is to speed half way through the roadway and slow down at the beginning/end of that roadway, while driving with an average speed elsewhere. However, the average speed and the variance will most likely differ among drivers and also depend on the characteristics of the roadways (i.e., average speed in highways is much higher than city roadways).

**Step 3: Prior probability.** Since we do not have enough information to model the target's route preferences, the probability that the target selects a particular route  $R_j^i$  (the prior probability) through a gap  $G_i$  is uniformly distributed for all available routes, and can be calculated as follows:

$$\Pr(R_j^i) = \frac{1}{f(G_i)}\quad (6.6)$$

---

<sup>3</sup>Here we assume that the target has a continuous motion, but if the target is suspected to have stopped en-route (e.g., to commit a crime), an estimated time of how long that could have lasted should be included in the calculation of  $t_{G_i}$ .

where  $f(G_i)$  is the total number of routes through the gap  $G_i$ . Although we assumed that taking any route is equally likely, in practice some routes are more likely to be taken by the target than others. This may be due to traffic flow conditions or POI for example, see section 6.5 for a discussion about such factors. Another way to model this is by adopting a mobility prediction algorithm, such as that presented in section 5.4, which predicts the direction a target vehicle would most likely take out an intersection by observing its current lane as it is approaching that intersection. However, running such algorithms is difficult on an offline tracking setting given our limited resources.

**Step 4: Marginal probability.** The marginal probability  $\Pr(t_{G_i})$  is the sum of the conditional (step 2) and prior (step 3) probabilities of all the routes and is calculated as follows (the marginal probability acts as a normalising constant in the sense that it makes sure that all the Bayesian probabilities of the routes will sum up to 1):

$$\Pr(t_{G_i}) = \sum_{j=1}^{f(G_i)} \Pr(t_{G_i}|R_j^i) \cdot \Pr(R_j^i) \quad (6.7)$$

**Step 5: Route selection.** Finally, we can now calculate the Bayesian probabilities for each route using equation 6.4 and select the route with the highest probability as the most likely route the target would have taken through the corresponding gap.

## 6.4 Simulation Results

In chapter 5, we proposed and simulated several tracking scenarios. In this chapter, we assume that such online tracking process has already been undertaken and that the collected tracking trace exhibits missing data. Since it is difficult to obtain real vehicular traces to validate our probabilistic trace reconstruction algorithm, we used a vehicular mobility simulator to generate artificial vehicular traces for different scenarios. Similar to the simulation in chapter 5, here we used VanetMobiSim simulator [54] to generate vehicular mobility traces based on IDM-LC (IDM with Lane Changes) mobility model [39] which is an extensions to the IDM (Intelligent Driver Motion) mobility model [125]. These traces are then fed into NS-2 simulator [69] to generate the movement of the entities (i.e., vehicles) and a trace database. Arbitrarily appointing one of the simulated entities as a target, we manually (and randomly) create gaps in that target’s mobility traces after extracting it from the original trace database. We then execute our trace reconstruction algorithm to probabilistically choose the most likely routes to connect those gaps and compare the selected routes with the routes the target actually took according to the original trace database. Since it is difficult to

model the delay factor  $\omega$  without having access to real traffic traces, we modelled  $\omega$  by observing the node density on the roadways that the gaps' routes are composed of. In particular, we refer to the original mobility traces for all the simulated nodes (before extracting the target's trace) and look at traces taken for any node that happen to be passing through any of the roadways that are part of the gap's routes, then estimate the average node density of these roadways to assign values for their corresponding  $\omega$  (clearly, the higher the node density, the larger the value of  $\omega$ ). This is easy to do in practice too since the records from which the target trace was extracted usually contain other information about other entities that we can use to model the delay factors, which then can be used in equations 6.2 and 6.3. However, note that different nodes may have different movement (driving) behaviours, so this modelling techniques will return just a rough estimate for  $\omega$ , but this is the best we can do without accessing external information (in section 6.3.2 we proposed technique based on such information).

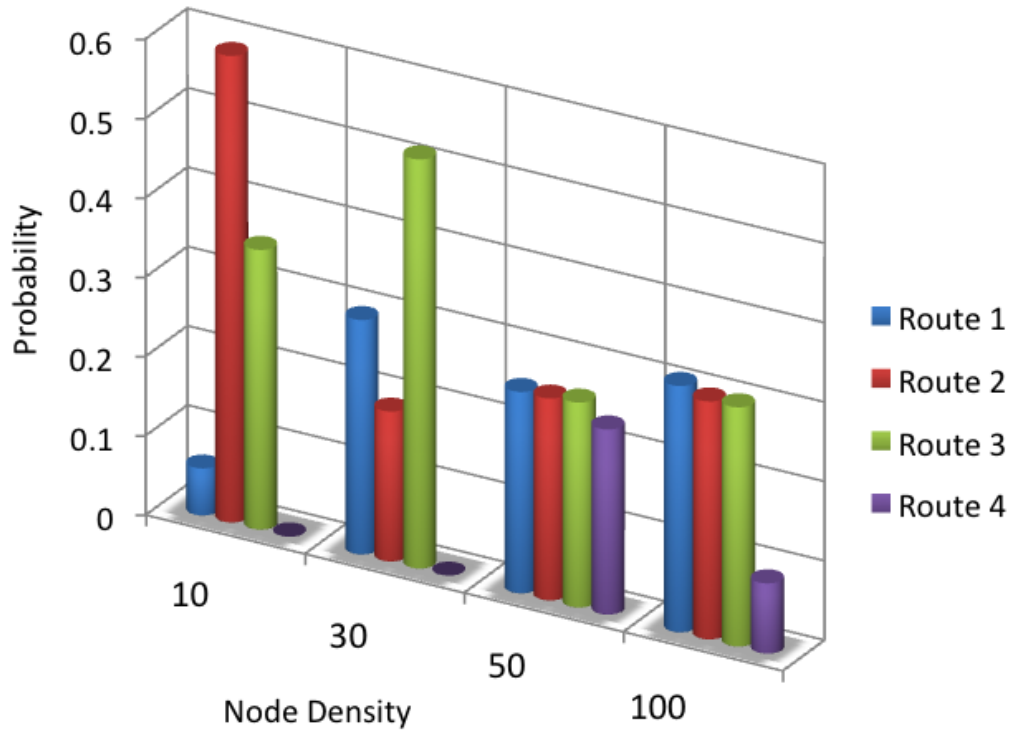


Figure 6.4: Simulation Results

Figure 6.4 illustrates our simulation results for scenarios with different node densities, ranging from 10 to 100 nodes, running over a 1000m<sup>2</sup> area consisting of roadways with different lengths. Every simulation was run for 1800 seconds and in each sce-

Simulation software	VanetMobiSim and NS-2
Simulation area	1000m <sup>2</sup>
Simulation duration	1800 seconds
Simulation runs	4
Node density	10–100
Node max. speed	varied
Mobility Model	IDM-LC
Communication technology	IEEE 802.11
Radio propagation model	Free space

Table 6.1: Simulation configurations

nario we manually created a gap (with 4 possible routes between the Ingress and the Egress) and run the algorithm to select the most probable route. Figure 6.4 shows the probabilities of each of the four possible routes at the manually created gap in each scenario. The results indicate that our reconstruction algorithm along with our modelling technique (by averaging the traffic flow of roadways) works very well in scenarios with lighter node densities where the probabilities of the routes vary drastically and it is easy to see which route is the most probable, but as node density increases, the probabilities become closer. Although we argue that this way of modelling  $\omega$  is, in most cases, efficient since it gives a good estimation of other traffic delay factors affecting the roadways without actually having to model them individually, further modelling may be required in the more cluttered scenarios. However, in all cases, there are generally some routes that can be easily ruled out (like route 4 in all scenarios) usually because they introduce much longer/shorter delay compared to the time the target trace was missing over the gap.

We believe that if these algorithms were applied to real traces, the results will be more accurate since most of the simulation-based mobility models (which we had to use) do not always maintain a tightly consistent motion (i.e., driving) behaviour for every entity, so modelling the driving behaviour of the target based on his history trace or traces from other nodes, although returning acceptable results, is slightly less accurate. In real life scenarios, on the other hand, every driver has a unique driving behaviour, in fact, recent work [129] even showed that the driving behaviour of individuals would make a reasonable biometric measure.

## 6.5 Offline Estimation Accuracy

The accuracy of our reconstruction algorithm is influenced by a number of factors, mainly concerning the accuracy of the tracking data collection. When collecting RF tracks, beside the conventional RF measurement errors, it is very likely that the tracking traces are collected by several entities, so unless these entities are tightly synchronised, there will be timing errors among the recorded traces. Similarly, the traffic delay factor  $\omega$  influences the accuracy of the algorithm, and is explicitly used in equations 6.2 and 6.3 while calculating the average speed and variance of the target. The delay factor  $\omega$  is basically a time delay assigned to individual roadways and is highly dependent on the various geographical and physical characteristics of the roadways. Examples of factors influencing  $\omega$  are (some of which were already discussed in section 5.6):

- *Roadways lengths*: routes are most likely composed of several roadways that are usually of different lengths. While the accumulation of the lengths of the roadways that form a route is representative to the distance between the Ingress and the Egress through that route, the speed of the movement through this route is affected by the accelerations and decelerations during the journey and around the intersections connecting the route's roadways. Thus, the speed of the target should be estimated for the individual roadway not the whole route.
- *Roadways speed limits*: every roadway restricts the speed of vehicles to a specific speed limit. Knowledge of these limits is useful for estimating the maximum time threshold during which a vehicle could pass the corresponding roadway.
- *Traffic management type*: traffic delay highly depends on the traffic management type. For example, it is very likely that an intersection managed by stop signs will experience longer delays than another managed by traffic lights. However, care should be taken when considering traffic lights because delays due to traffic lights depend on the state of the traffic light upon arrival (i.e., vehicles reaching the intersection while the traffic light indicates green will most likely experience much less delay than otherwise), which is generally difficult to model accurately.
- *Points of Interest (POI)*: Another very influencing factor is the existence of points of interests. These points represent locations that are frequently visited by people, such as banks, shops etc., and hence are locations of common interest. Intuitively, the existence of such points along a roadway will very likely increase the traffic density at that roadway and, consequently, the traffic delay. Potentially, information about POI can be extracted from a few specialised maps (like Google maps) and then used to derive a more accurate prior probability in equation 6.6.

- *Traffic density and flow*: if available, knowledge of vehicular density (number of vehicles per *km*) and flow (number of vehicles crossing a point per hour) is very useful. Such information can either be statistically estimated from empirical data, or probabilistically inferred based on location and time. For example, a particular area may be increasingly crowded/congested only during a particular period of time in the day, such as an intersection leading to offices, which may be congested only at early morning and late afternoon.
- *Abnormal events*: occasionally, abnormal events may occur and will accordingly affect the traffic conditions in the corresponding roadways. Examples of such (temporary) abnormal events include roadwork, accidents, emergencies etc. In most cases, information about such events can be obtained from the police.

Clearly, obtaining information to accurately model the traffic factor  $\omega$  is difficult, this is why in sections 6.3.2 and 6.4, we proposed techniques to model  $\omega$  without having to access extra information about the actual traffic. In particular, we estimate the traffic flow of the concerned roadways by referring to the original tracking traces (before extracting the target's trace off). However, beside the explicit modelling of  $\omega$  as described above, we note that we already implicitly (and partially) account for it following our modelling techniques in sections 6.3.2 and 6.4. That is, when we averaged the driving times in step 1 (section 6.3.2), we have already implicitly accounted for  $\omega$  since the traces we used in calculating the average driving times already included an implicit  $\omega$ . This will probably work most of the time, but modelling  $\omega$  separately is certainly preferred if adequate resources are available.

## 6.6 Summary

In this chapter, we considered an offline vehicular tracking scenario where target traces are obtained and probabilistically reconstructed. This is in contrast to online vehicular tracking (which was discussed in chapter 4 for pedestrian setting, and in chapter 5 for vehicular setting) that tracks targets in real time. Since the tracking gaps in a target trace may be prohibitively large, we introduced preparatory phase that try to minimise these gaps by fusing tracks from different sources. Once the trace is prepared, the trace reconstruction algorithm proceeds in two phases. In the first phase the end points of the gaps along with the possible routes through the gaps are obtained. Using Bayesian inference, the most probable routes through these gaps are then reconstructed. We also discussed the various factors influencing the accuracy of this estimation process and presented some simulation results.

## Chapter 7

# Offline Multi-modal Forensic Tracking

*Most contemporary civilian tracking applications consider an online approach where the target is being tracked in real time. In criminal investigations, however, it is common that only offline tracking is possible, where tracking takes place after the fact. In this case, given an incomplete trace of a target (suspect), the task is to reconstruct the missing parts and obtain the full trace. With the recent proliferation of modern transportation systems, target entities are likely to interact with different transportation means. Thus, in this chapter, we first introduce a class of mobility models that has been especially tailored for forensic analysis and propose several instances emulating different transportation means. We then use these models to build a fully-fledged offline multi-modal forensic tracking system that reconstructs an incomplete trace of a particular target. We provide theoretical evaluation of the reconstruction algorithm and show that it is both complete and optimal. The contents of this chapter was published in [8] and [9].*

### 7.1 Introduction

Traditional digital forensics was exclusively concerned with extracting evidence and traces from electronic devices that may have been associated with or used in a criminal activity. In most criminal cases, however, it would also be desirable to find additional information about particular suspects, such as their physical activities (not just the digital ones). In particular, investigating the location of suspects before, during and after a crime, may contribute significant evidence, especially if it was possible to prove

that a suspect was in a particular location at a particular time that he previously denies. This kind of investigations is called *forensic tracking*, which we introduced and discussed extensively in chapter 3. In most criminal cases, forensic tracking is carried out in an offline manner, where a location *trace* of a suspect is obtained and probabilistically reconstructed to recover any missing parts; such trace can be of a target that is randomly captured by several CCTV cameras scattered over a particular area. In chapter 6, we discussed how to carry out such offline forensic investigation in a vehicular setting. In this chapter, we extend this further and consider a multi-modal<sup>1</sup> environment. In particular, we propose a generic trace reconstruction framework and adopt it to build a complete multi-modal-based forensic tracking system.

We assume that we have access to incomplete location information of a target showing when and where he was observed; this will create a set of scattered points over an area. We then need to *connect* these points to be able to find out what routes the target could have taken (this is illustrated in figure 7.1). These periods of missing data (between the points where the target was observed) are called *gaps*, which if we reconstruct properly, we can obtain the target’s full trace. Since we generally need to evaluate all possible routes through these gaps in a multi-modal scenario, we will need to consider pedestrian routes, public routes, and a combination of both. Briefly, our trace reconstruction algorithm proceeds in three main phases (we describe how the algorithm reconstructs a trace belonging to a single target, the same process is repeated to reconstruct the traces of multiple targets):

1. *Scene representation*: in this phase a map of the crime scene and its surrounding area is obtained. Points corresponding to special locations, such as the crime location, the available traces of the target (where the target was observed), and the transport routes are then marked.
2. *Fuzzy validation*: this is a complementary phase where the (incomplete) trace of the target obtained in the previous phase is further validated using fuzzy logic.
3. *Trace reconstruction*: once the scene has been graphically represented and prepared, the reconstruction algorithm is executed over the gaps. Each gap represents a period of missing location information and is reconstructed independently. All the reconstructed gaps are connected to obtain the most probable route(s) the target most likely has taken through these gaps.

**Chapter Outline.** This chapter is organised as follows. In section 7.2 we propose a generic trace reconstruction framework, followed by a description of how a scene is rep-

<sup>1</sup>Multi-modal refers to a journey involving more than one mode of transportation.

resented graphically in preparation of the reconstruction process in section 7.3. Several (special-purpose) mobility models are then proposed in section 7.4, which are later used by our trace reconstruction algorithm in section 7.6. Prior to actually reconstructing the trace, we use fuzzy logic in section 7.5 to validate the existing traces, which is important (but optional) since some of the available traces (which our reconstruction process is based on) may have been generated unreliably.

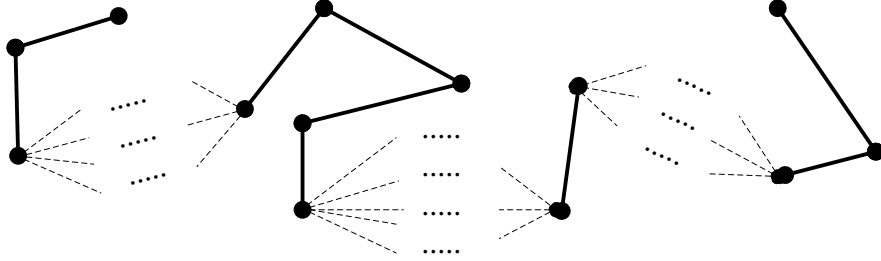


Figure 7.1: Sample target trace (with gaps)

## 7.2 Trace Reconstruction Framework

Figure 7.1 depicts a common scenario in most criminal investigations dealing with suspect trace reconstruction. In this scenario, the suspect has been observed at some locations, but was unobservable at others. The problem is how to use the available traces (the observed locations) to reconstruct the missing ones. Suppose we have a trace  $T_s = V_s + W_s$  for a target  $s$ , where  $V_s$  and  $W_s$  are the available and missing parts of  $s$ ' trace, respectively. The  $i$ -th gap between the points  $a$  and  $b$  in  $T_s$  is denoted by  $G_i^{a,b} \in W_s$  and represents a period of missing observation. In each  $G_i^{a,b}$ , there are  $f(G_i)$  possible routes between  $a$  and  $b$  (where  $f$  here returns the number of routes connecting a gap, as defined in section 6.3.2). The reconstructed trace  $T'_s = V_s + W'_s$  then consists of the available traces  $V_s$  and the most probable routes  $W'_s$  connecting the gaps. Formally, we propose and adopt a reconstruction framework that proceeds as follows (algorithm 7.1 provides a pseudocode):

1. Obtain a map  $M$  of the crime scene and the surrounding areas, and convert it to a graph  $G_M$ .
2. Obtain the original trace  $T_s$  of the target  $s$  (with  $n$  gaps), and annotate the gaps  $G_i^{a,b}$  where  $i \in \{1, 2, \dots, n\}$ , and  $a, b$  are the end points of the  $i$ -th gap.
3. For each  $G_i^{a,b}$ , find all possible connecting routes  $R_j$  where  $j \in \{1, 2, \dots, f(G_i)\}$  and  $f(G_i)$  is the number of routes connecting  $a$  and  $b$ .

4. For each  $R_j$ , obtained in step 3, reason about whether or not the target  $s$  could have taken  $R_j$  while traversing the gap  $G_i^{a,b}$ , we do this by assigning probabilities/costs to each  $R_j$ ; repeat for all gaps.
5. The reconstructed trace  $T'_s$  is then formed by connecting the available parts of the trace with the  $R_j$ 's that have been assigned the highest probability in step 4.

---

**Algorithm 7.1** Trace Reconstruction Framework
 

---

```

1: GET  $M$  {obtain the scene map}
2: GET  $T$  {obtain the target's trace with missing data}
3: convertToGraph( $M$ )  $\rightarrow G$ 
4: mark( $T, G$ ) {mark  $T$  on top of  $G$ }
5: findGaps( $G$ )  $\rightarrow G.gap$  {find the gaps in  $G$ }
6: for  $i = 1$  to  $i \leq \text{sizeof}(G.gap)$  do
7:   getRoutes( $G.gap[i]$ ) =  $G.gap[i].R$  {store routes through  $G.gap[i]$  in array  $R$ }
8:   for  $j = 1$  to  $j \leq \text{sizeof}(G.gap[i].R)$  do
9:     assignProbability( $G.gap[i].R[j]$ )
10:  end for
11:  findHighestProbability( $G.gap[i].R$ )  $\rightarrow G.gap[i].\text{mostProbableRoute}$ 
12: end for
13: fillGaps( $G$ ) {connect the gaps}
14: return reconstruct( $G$ ) {return the full trace}

```

---



---

**Procedure 7.2** Gap Filling Procedure
 

---

```

1: fillGaps( $G$ )
2: for  $i = 1$  to  $i \leq \text{sizeof}(G.gap)$  do
3:   connect( $G.gap[i].\text{Ingress}, G.gap[i].\text{Egress}$ )  $\rightarrow G.gap[i].\text{mostProbableRoute}$ 
   {connect the end points of a gap with the most probable route through it}
4: end for

```

---

In step 1, it is trivial to obtain maps from public sources. Converting a map  $M$  to a graph  $G_M$ , though, might require extra information, which may still be available publicly. The graph  $G_M$  would consist of a set of vertices  $\mathcal{V}$  and a set of edges  $\mathcal{E}$ . In this chapter we consider a multi-modal scenario, where the target uses different means of transportation, and thus we need to have more than one type of edges and vertices to distinguish between the different transportation modes; this edge/vertex distinction will allow us later to correctly assign probability/cost weights to the routes. If we have access to some additional information about the scene, such as locations of points of interest (POI), estimated traffic flow etc., we may also include these in the generated graph by marking them appropriately; this information will be useful when we reason about the likelihood of the routes in step 4. Step 2 is trivial. In step

3, a special route-counter algorithm is adopted to find the possible routes through the gaps. Clearly, finding *all* possible routes may require an exponential time (for relatively large gaps/scenario), thus the search area should first be bounded before executing the route-counter algorithm, an example of such algorithm is BRC which we proposed in chapter 6 (a more advanced algorithm covering wider scenario space is proposed in section 7.6.1). Step 4 is the heart of the reconstruction algorithm where the most probable routes through the gaps are selected. Finally, in step 5 we connect all routes chosen on step 4 with the available parts of the trace to obtain the final reconstructed trace.

In the rest of the chapter, we adopt this framework and propose an offline multi-modal tracking system, which can be considered a practical instantiation of the framework. Steps 1-3 are covered in section 7.3 while steps 4-5 are covered in section 7.6.

### 7.3 Scene Representation

In order to systematically reconstruct the target’s trace, a graphical representation of the crime scene and the surrounding area has to be first generated. Figure 7.2 graphically illustrates the scene preparation process over a sample map<sup>2</sup>, and algorithm 7.3 provides a pseudocode. The scene representation algorithm proceeds in 5 steps as follows (to simplify the notation, we will often drop unnecessary labels and tags while referring to some edges and vertices in the map):

**Step 1: Map Preparation.** In this initial step, a schematic map  $G_M$  (based on an actual geographical area  $M$ ) of the reconstruction scene (the area over which the target trace needs to be reconstructed) is obtained. We do not impose any restrictions on the size of  $G_M$  other than requiring it to at least cover (1) all the points at which the target was observed (the available trace of the target), and (2) the crime location(s). Formally, let  $G_M = (\mathcal{V}^{G_M}, \mathcal{E}^{G_M})$  be the scene graph, where  $\mathcal{V}^{G_M}$  and  $\mathcal{E}^{G_M}$  are the sets of vertices and edges of  $G_M$ , then we assume that  $\{X_s^{G_M} \cup \hat{C}^{G_M}\} \in \mathcal{V}^{G_M}$ , such that:

1.  $X_s^{G_M} = \{x_1^{\kappa_p}, \dots, x_n^{\kappa_q}\}$  is the set of all locations where the target  $s$  was observed at, where  $\kappa_p < \kappa_q$  are the first and last time, respectively,  $s$  was observed in  $G_M$ ,
2.  $\hat{C}^{G_M} = \{c_1^{\kappa_k}, \dots, c_m^{\kappa_l}\}$  is the set of several crime locations where crimes were committed between times  $k$  and  $l$ . However, to simplify the discussion, we will describe the reconstruction algorithm considering a single crime scenario, but the algorithm is obviously applicable to multiple crimes scenarios too.

<sup>2</sup>While the map in figure 7.2 corresponds to a real map of the city of Boston of the United States of America, the road network in figure 7.2(d) is artificial, it is plotted for illustrative purposes only.

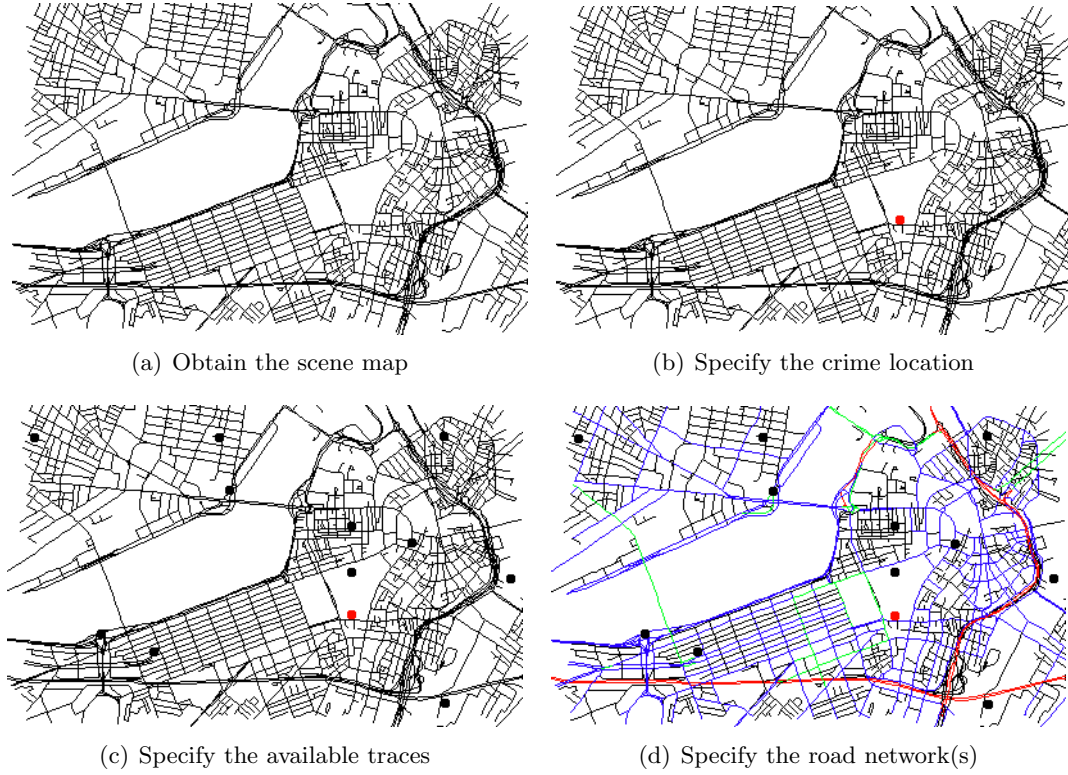


Figure 7.2: Illustration of the scene preparation phase

**Step 2: Route Marking.** In this step, relevant public transport networks (e.g. buses, trains)  $B_1, B_2, \dots, B_n \in \mathcal{B}$  are marked on  $G_M$ . A transport network  $B_j \in \mathcal{B}$  consists of a set of routes  $B_j = \{R_1^{B_j}, R_2^{B_j}, \dots, R_r^{B_j}\}$ , which constitute most of the vertices and edges in  $G_M$ . Since we are only marking public transport routes, vertices in a route  $R_i$  correspond to either a stop (e.g. bus/train station), denoted  $S$ -vertex, or a road turn, denoted  $U$ -vertex. Similarly, edges can either be routed (part of a route), denoted  $R$ -edge, or unrouted, denoted  $W$ -edge (the latter mostly added in Step 4). Let  $e^i$  be an edge of type  $i$ , then:

$$i = \begin{cases} B & \text{if } e \in \bigcup_{B_j \in \mathcal{B}} \bigcup_{i=1}^{|B_j|} R_i^{B_j} \\ W & \text{if } e \in \mathcal{E}^{G_M} \setminus \bigcup_{B_j \in \mathcal{B}} \bigcup_{i=1}^{|B_j|} R_i^{B_j}. \end{cases}$$

where the notation  $|x|$  means the number of elements in the set  $x$  (the length of  $x$ ), assuming that  $x$  does not have repeated elements (i.e., correspond to loop-free routes). A route  $R_i$  is, therefore, defined by the set of vertices it consists of,  $V_{R_i} = \{v_1, v_2, \dots, v_k\}$ , and the edges linking these vertices  $E_{R_i} = \{e_1, e_2, \dots, e_{k-1}\}$ . Once all routes are marked, we plot the available trace of the target,  $X_s^{G_M} = \{x_1, x_2, \dots, x_n\}$ , which spec-

**Algorithm 7.3** Scene Representation

---

```

1: GET  $M$  {obtain the crime scene map (to be converted to  $G$ )}
2: GET  $X$  {obtain the locations where the target was observed}
3: GET  $C$  {obtain the crime locations}
4: GET  $B_1, B_2, \dots, B_n$  {obtain the transport networks}
5: for  $i = 1$  to  $i = n$  do
6:   getAllVertices( $B_i$ )  $\rightarrow$  append( $V$ ) {store all the vertices of all the routes in  $V$ }
7:   getAllEdges( $B_i$ )  $\rightarrow$  append( $E$ ) {store all the edges of all the routes in  $E$ }
8: end for
   Step 2: Route Marking
9: for  $i = 1$  to  $i = n$  do
10:  getRoutes( $B_i$ )  $\rightarrow R_i[\text{sizeof}(B_i)]$  {store the routes of  $B_i$  at the array  $R_i$ }
11:  markRoutes( $R_i$ ) {Call the the markRoute procedure}
12: end for
   Step 3: Vertex/Edge Labelling
13: labelling( $V, E$ ) {call the labelling procedure }
   Step 4: End Vertices
14: for  $i = 1$  to  $n$  do
15:  getRoutes( $B_i$ )  $\rightarrow R_i[\text{sizeof}(B_i)]$  {store the routes of  $B_i$  at the array  $R_i$ }
16:  endVertices( $R_i$ )  $\rightarrow \rho$  {call the endVertices procedure below}
17: end for
   Step 5: Additional Edges
18: SET  $W_{max}$  {a distance threshold used by the additionalEdges procedure}
19: additionalEdges( $V, W_{max}$ ) {call the additionalEdges procedure}

```

---

ify the times and locations where the target was observed in  $G_M$  (these will later form the gaps that we need to reconstruct). All  $x_i$ 's are either<sup>3</sup> located on top of vertices or over edges (corresponding to locations on road or at intersection), that is  $x_1, x_2, \dots, x_n \in \mathcal{V}^{G_M} \cup \mathcal{E}^{G_M}$ . However, elements in  $X_s^{G_M}$  should naturally be represented as vertices, thus if any  $x_i$  is located on  $e_i \in \mathcal{E}^{G_M}$ ,  $e_i$  is split at the location of  $x_i$  such that  $e_i = e_i^1 + e_i^2$ . Then,  $x_i$  is added to  $\mathcal{V}^{G_M}$  (as  $U$ -vertex) and  $e_i$  is replaced by  $e_i^1, e_i^2$  in  $\mathcal{E}^{G_M}$ , while updating the  $V^{R_i}$  and  $E^{R_i}$  (the sets of all vertices and edges in  $R_i$ ) of any route  $R_i$  passing through  $e_i$ . Next,  $\hat{C}^{G_M} = \{c_1, c_2, \dots, c_m\}$ , the locations of the crimes, are marked on  $G_M$ , but this time  $c_1, c_2, \dots, c_m$  may not be on top of a vertex or an edge, in which case a  $W$ -edge is created between  $c_i$  and the closest  $v_i \in \mathcal{V}^{G_M}$ . Notice that it is acceptable for  $c_i$  to be on top of an edge  $e_i \in \mathcal{E}^{G_M}$  because  $\hat{C}^{G_M}$  will not be involved in the reconstruction process. Finally, the directions of all edges  $e_i \in \mathcal{E}^{G_M}$  are specified. The directions of  $e_i^R$  ( $R$ -edges) can easily be determined by referring to their corresponding routes, while  $e_i^W$  ( $W$ -edges) are undirected.

---

<sup>3</sup>We assume that  $X_s^{G_M}$  data was collected by roadside CCTV cameras, and for simplicity, we assume that the locations of  $x_1, \dots, x_n$  correspond to the locations of the CCTV cameras that recorded them.

Generally, a single edge  $e_i$  or vertex  $v_i$  cannot have two different types at the same time. If a particular vertex  $v_i$  is part of  $n$  routes  $R_i, R_j, \dots, R_n$ , then it is an  $S$ -vertex as long as  $v_i$  is an  $S$ -vertex in at least one of  $R_i, R_j, \dots, R_n$ , otherwise it is  $U$ -vertex. On the contrary, edges are not allowed to be part of more than one route because different routes may assign different weights to their edges (see section 7.4 for how and when these weights are assigned). Thus, if there is more than one route traversing an edge, we create as many edges as there are routes. Let  $e_i : v_p \rightarrow v_q$  be an edge between vertices  $v_p$  and  $v_q$ , and suppose there are  $n$  routes passing through  $e_i$ , then we relabel  $e_i$  to  $e_{i,1}$  and create  $n-1$  extra edges and label them  $e_{i,2}, \dots, e_{i,n}$ . Thus,  $G_M$  is a mixed multi-graph; that is, a graph allowing multiple directed edges to have the same head and tail vertices, and contains both directed ( $R$ -edge) and undirected ( $W$ -edge) edges.

---

**Procedure 7.4** Route Marking Procedure
 

---

```

1: markRoutes( $R$ )
2: for  $i = 1$  to  $i = \text{sizeof}(R)$  do
3:   getVertices( $R_i$ )  $\rightarrow V_i[\text{sizeof}(R_i)+1]$  {obtain the vertices of each route; each route
   consists of  $n$  roadways (edges) and  $n+1$  vertices (intersection)}
4:   getEdges( $R_i$ )  $\rightarrow E_i[\text{sizeof}(R_i)]$  {obtain the edges of each route}
5:   for  $j = 1$  to  $j = \text{sizeof}(V_i)$  do
6:     if  $V_i[j] = \text{STOP}$  then
7:        $V_i.\text{type} = S$  {if the vertex corresponds to a stop, it is  $S$ -vertex}
8:     else if  $V_i[j] = \text{TURN}$  then
9:        $V_i[j].\text{type} = U$  {if the vertex corresponds to a turn, it is  $U$ -vertex}
10:    end if
11:  end for
12:  for  $j = 1$  to  $j = \text{sizeof}(E_i)$  do
13:    if  $E_i[j] = \text{ROUTED}$  then
14:       $E_i[j].\text{type} = R$  {if the edge is part of a route, it is  $R$ -edge}
15:    else if  $E_i[j] = \text{UNROUTED}$  then
16:       $E_i[k].\text{type} = W$  {if the edge is not part of a route, it is  $W$ -edge}
17:    end if
18:  end for
19: end for{end of  $R$ 's route loop}

```

---

**Step 3: Vertex/Edge Labelling.** All vertices and edges (except  $W$ -edges) are assigned unique labels to specify the routes they are part of. The notation of a vertex  $v_i^{\ell_i}$  with label  $\ell_i = R_j^k, \dots, R_m^n$  indicates that the  $i$ -th vertex in  $\mathcal{V}^{G_M}$  is simultaneously the  $k$ -th,  $\dots$ ,  $n$ -th vertex of routes  $R_j, \dots, R_m$ , respectively. Since all vertices are part of routes, a label  $\ell$  should contain information about at least one route. Edges are characterised by the vertices they link, thus the notation  $e_i^{\ell_i} : v_p^{\ell_p} \rightarrow v_q^{\ell_q}$  means

that the  $i$ -th edge in  $\mathcal{E}^{G_M}$  has head and tail at  $v_p, v_q \in \mathcal{V}^{G_M}$ , respectively, where  $p, q \in \{1, 2, \dots, |\mathcal{V}^{G_M}|\}$ . Note that the head  $v_p$  and tail  $v_q$  should belong to at least one common route and are ordered in succession according to the direction of the edge. If there is more than one route passing through  $e_i$ , extra parallel edges are created and labeled, as detailed in Step 2. Thus, the label of an edge has only one route.

---

**Procedure 7.5** Vertex/Edge Labelling Procedure
 

---

```

1: labeling( $V, E$ )
2: for  $i = 1$  to  $\text{sizeof}(E)$  do {Edge labelling}
3:   if  $E[i].\text{type} = S$  then
4:      $E[i].\text{label} = E[i].\text{route}$  {label  $E[i]$  according to the routes it is part of}
5:   else
6:      $E[i].\text{label} = \perp$  {otherwise, it must be a  $W$ -route, which is unlabelled}
7:   end if
8: end for
9: for  $i = 1$  to  $i = \text{sizeof}(V)$  do {Vertex Labelling}
10:  for  $j = 1$  to  $\text{sizeof}(E)$  do
11:    if  $E[j].\text{head} = V[i]$  or  $E[j].\text{tail} = V[i]$  then
12:       $V[i].\text{label} = \text{append}(E[j].\text{route})$  {add a route to a vertex label}
13:    end if
14:    {Unlike edges, vertices can be part of more than 1 route}
15:  end for
16: end for

```

---

**Step 4: End Vertices.** Once all vertices and edges are labelled, a special set  $\rho^{G_M}$  is created containing all *end* vertices, these are the first and last vertices of every route  $R_i \in B_j$  (the head and tail of  $R_i$ ). To simplify the discussion, we will consider routes of a single transport network  $B_j$ , but this can easily be extended to multiple networks  $B_m, \dots, B_n \in \mathcal{B}$ . Vertices belonging to  $\rho^{G_M}$  are found by first writing down the adjacency matrices  $A^{R_1}, A^{R_2}, \dots, A^{R_n}$  of all the routes  $R_1, R_2, \dots, R_n \in B_j$ , where  $f(B_j) = n$  ( $B_j$  contains  $n$  routes). Adjacency matrices represent the neighbouring relations among the vertices, if vertex  $i$  is adjacent to vertex  $j$ , then the cell  $A_{i,j}$  in the matrix  $A$  is set to 1, 0 otherwise. A particular vertex in  $R_i$  belongs to  $\rho^{G_M}$  if its corresponding row in  $A^{R_i}$  sums up to 1. We denote by  $A_{i,j}^{R_i}$  the element of the  $i$ -th row and  $j$ -th column of  $A^{R_i}$ , if we need to refer for a whole row, we use the notation  $A_{i,*}^{R_i}$ , and similarly denote a whole column by  $A_{*,j}^{R_i}$  (i.e.,  $A^{R_i} = A_{*,*}^{R_i}$ ). Thus, formally:

$$\rho^{G_M} = \bigcup_{B_j \in \mathcal{B}} \rho_{B_j}^{G_M} = \bigcup_{B_j \in \mathcal{B}} \bigcup_{R_i \in B_j} \left( v_k : \sum_{k \in R_i} A_{k,*}^{R_i} = 1 \right)$$

Figure 7.3 shows a graph consisting of routes  $R_1, R_2, R_3$  in which we illustrate how

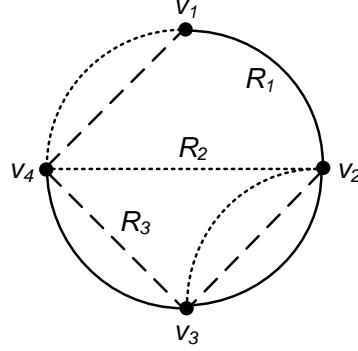


Figure 7.3: A graph of three routes  $R_1, R_2, R_3$

to construct adjacency matrices for. Basically, adjacency matrices efficiently describe the neighbouring relations between the vertices in a graph. For example, vertex  $v_1$  in  $R_1$  is adjacent (neighbour) to vertex  $v_2$ , so  $A_{1,2}^{R_1} = 1$ . Note that we keep the notion of self-neighbour undefined, that is a vertex cannot be a neighbour to itself, so the diagonal in the matrices is necessarily always filled with 0's.

$$\begin{array}{c}
 \begin{array}{cccc}
 v_1 & v_2 & v_3 & v_4 \\
 v_1 \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \\
 A^{R_1}
 \end{array}
 &
 \begin{array}{cccc}
 v_1 & v_2 & v_3 & v_4 \\
 v_2 \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \\
 A^{R_2}
 \end{array}
 &
 \begin{array}{cccc}
 v_1 & v_2 & v_3 & v_4 \\
 v_3 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \\
 A^{R_3}
 \end{array}
 \end{array}$$

**Proposition 7.3.1.** *A vertex  $v_j \in V_i$  in the adjacency matrix  $A^{R_i}$  of a finite loop-free route (i.e., simple path)  $R_i = (V_i, E_i)$ , where  $V_i$  and  $E_i$  are the sets of vertices and edges forming  $R_i$ , is an end vertex if its corresponding row  $A_{j,*}^{R_i}$  in  $A^{R_i}$ , sums up to 1.*

*Proof.* Let the route  $R_i$  be represented by the ordered sequence of vertices  $v_1, v_2, \dots, v_n$ , where  $v_1$  and  $v_n$  are the first and last vertices of  $R_i$ , these are called the end vertices. Clearly,  $v_1$  and  $v_n$  will each be adjacent to a single vertex belonging to  $R_i$ , namely  $v_2$  and  $v_{n-1}$ , respectively. All other vertices,  $v_2, \dots, v_{n-1}$  are adjacent to two vertices belonging to  $R_i$ ; that is  $v_i$  is adjacent with  $v_{i-1}$  and  $v_{i+1}$ , for  $i \in \{2, \dots, n-1\}$ . Therefore,  $\sum A_{1,*}^{R_i} = \sum A_{n,*}^{R_i} = 1$ , while  $\sum A_{i,*}^{R_i} = 2$  for  $i \in \{2, 3, \dots, n-1\}$ .  $\square$

Since routes in  $G_M$  are directed,  $\rho^{G_M} = \overrightarrow{\rho}^{G_M} \cup \overleftarrow{\rho}^{G_M}$ , where  $\overrightarrow{\rho}^{G_M}$  and  $\overleftarrow{\rho}^{G_M}$  are the sets of head and tail end vertices of the routes in  $G_M$ .

**Procedure 7.6** End Vertices Procedure

---

```

1: endVertices( $R$ )
2: for  $i = 1$  to  $\text{sizeof}(R)$  do
3:    $\text{getVertices}(R_i) \rightarrow V$ 
4:   for  $i = 1$  to  $i = \text{sizeof}(V)$  do
5:     for  $j = 1$  to  $i = \text{sizeof}(V)$  do
6:       if  $\text{neighbours}(V[i], V[j]) = \text{True}$  then
7:          $V[i].\text{routeCount} = + 1$  {increment the router counter}
8:       end if
9:     end for
10:    if  $V[i].\text{routeCount} = 1$  then
11:       $\rho \leftarrow \text{add}(V[i])$  {add  $V[i]$  to the set of end vertices  $\rho$ }
12:    end if {if  $\text{routeCount} = 1$ , then it must be either a head or a tail of a route}
13:  end for
14: end for

```

---

**Step 5: Additional edges.** In this final step we create additional  $W$ -edges between several vertices. A new  $W$ -edge is created between any two  $S$ -vertices if: (1) they belong to different routes, and (2) the distance between them is  $\leq W_{max}$  (a particular threshold). Formally, the set  $\eta$  of the new (undirected)  $W$ -edges is:

$$\eta = \left\{ e_k^{W, \ell_k} = v_m^{S, \ell_m} \leftrightarrow v_n^{S, \ell_n} : \ell_m \neq \ell_n \wedge d(v_m^{S, \ell_m}, v_n^{S, \ell_n}) \leq W_{max} \right\}$$

where  $\leftrightarrow$  represents an undirected edge, and  $d(x, y)$  is the distance between  $x$  and  $y$ . Note that here we disregard the effect of the infrastructure on the  $W$ -edges, that is, we assume that there are no major obstacles between the  $S$ -vertices that prevent  $W$ -edges from being created. However, integrating infrastructure information is easy since most modern maps contain such information, where we can find  $d(x, y)$  by rerouting around the infrastructure and test whether it is  $\leq W_{max}$ . Finally, we can now formally define the graph  $G_M = (\mathcal{V}^{G_M}, \mathcal{E}^{G_M})$  in terms of its edges and vertices, as  $\mathcal{E}^{G_M} = E_R^{G_M} \cup E_W^{G_M}$  and  $\mathcal{V}^{G_M} = X_s^{G_M} \cup \hat{C}^{G_M} \cup V_S^{G_M} \cup V_U^{G_M}$ , where  $E_X^{G_M}$  and  $V_X^{G_M}$  are the sets of  $X$ -edges and  $X$ -vertices in  $G_M$ , respectively.

## 7.4 Mobility Modelling

Mobility models were traditionally mainly used in computer simulation, where running an experiment (e.g. evaluating a protocol) on real systems is both costly and inconvenient. Basically, mobility models generate artificial mobility traces that ideally resemble mobility patterns of real entities. These traces, nevertheless, cannot be directly used to reconstruct real traces that have already been made by real-life entities. This is mainly

**Procedure 7.7** Additional Edges Formation Procedure

---

```

1: additionalEdges( $V, W_{max}$ )
2: for  $i = 1$  to  $\text{sizeof}(V)$  do
3:   for  $j = 1$  to  $\text{sizeof}(V)$  do
4:     if  $V[i].\text{type} = S$  and  $\text{distance}(V[i], V[j]) \leq W_{max}$  then
5:       createW-Route( $V[i], V[j]$ )
6:     else if  $V[i].\text{type} = S$  and  $V[i].\text{label} \neq V[j].\text{label}$  then
7:       createW-Route( $V[i], V[j]$ )
8:     end if
9:   end for
10: end for

```

---

because real mobility patterns are based on human judgements, which are usually very stochastic in nature. However, we show that even though mobility traces generated by these models are inherently artificial, we can still use them effectively to assist our reconstruction process.

Mobility models are usually developed in a microscopic level, modelling the mobility of each object in relation to its environment and the surrounding neighbouring objects, and thus generating realistically-looking traces. Most models, therefore, carefully parameterise the velocity (or rather the acceleration) and direction of the objects and repeatedly adjust them throughout the simulation. However, in our case, we only need to use the mobility models to estimate the time a target entity may have spent while moving from one point to another, completely disregarding the precise microscopic details; we call this class of mobility models *mobility delay models*. In addition, since we are considering a multi-modal scenario (an entity occasionally changing transportation mode), we not only need to model each class of mobility, but also need to model how to *glue* different models together for a smooth transition. We now introduce several mobility delay models, and then proceed to model the transition between them.

#### 7.4.1 Pedestrian Mobility Delay Model

Popular pedestrian mobility models, such as the social force model [59], cannot be directly used in our scenario because it requires detailed microscopic information which we cannot assume to possess (see section 3.4.1), it also delves into details of the inter-pedestrians behaviour, which is not important in our case. We, therefore, introduce a mobility delay model, which we call (Pedestrian Mobility Delay Model) PMDM, to calculate the time an entity  $x$  (pedestrian) would take to move from point  $a$  to point  $b$  (since here we are only concerned about the time). The mobility of the entity is mainly influenced by the surrounding static and moving objects which force the

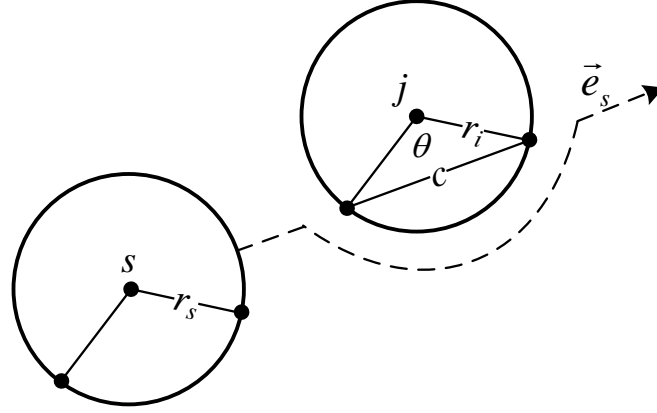


Figure 7.4: Illustration of the manoeuvre behaviour of a target and how to calculate the extra distance

entity to perform a suitable *manoeuvre* in order to avoid them. We represent each obstacle object (that the entity has to avoid) as a circle with known centre and radius. When manoeuvring around an obstacle, the extra distance the entity has to travel is approximately the length of the arc formed by the chord cutting through the obstacle's circle based on the entity's direction, this is illustrated in figure 7.4. The time the entity spends from point  $a$  to point  $b$  is then calculated as follows:

$$t_{a,b}^s = \frac{d_{a,b} + \sum_{i \in \mathcal{S}} r_i \cos^{-1} \left( \frac{2r_i^2 - c^2}{4r_i} \right) + \omega}{v_s - \epsilon} + \sum_{j \in \mathcal{M}} g((r_s + r_j) - d_{s,j}) \quad (7.1)$$

where  $\mathcal{S}$  and  $\mathcal{M}$  are the spaces of the static and moving objects, respectively (i.e., the number of objects in the scenario),  $r_i$  is the radius of the circle surrounding an object (representing its range),  $c$  is the length of the chord cutting through the object's circle (can be obtained via secant line geometry and the direction  $\vec{e}_s$  of the target),  $r_s$  is the radius of the circle surrounding the target  $s$  who moves with a speed up to  $v_s$ ,  $d_{s,j}$  is the distance between the centre of  $s$  and the centre of  $j$ ,  $\omega$  is a slight delay due to random factors imposed on the entity, such as crossing a road,  $\epsilon$  is a random negative value (modelling the deceleration behaviour the entity is forced to undertake around obstacles), and  $g()$  is a function defined as follows:

$$g((r_s + r_j) - d_{s,j}) = \begin{cases} 1 & \text{if } r_s + r_j \geq d_{s,j}, \\ 0 & \text{otherwise.} \end{cases}$$

which models the entity's pause time should it come across a moving object (waiting for it to move away). At first glance, equation 7.1 may seem to include microscopic

details since it models interactions between objects, but we note that we can model these details without necessarily simulating the scenario in a microscopic level and only by assuming knowledge of the movement directions of the entity. The manoeuvre behaviour of the entity around static objects (e.g. buildings) can then be easily modelled by referring to the scene map  $G_M$ , while the number of interactions between an entity and the moving objects (i.e., other pedestrians) can be estimated subjectively based on the popularity of the area and the current time of the day.

### 7.4.2 Transport Mobility Delay Models

Another class of common mobility models describes the mobility of vehicular entities, modelling public transport means, such as buses, trains, trams, underground tubes etc. (here we do not consider private vehicles because they are irrelevant to our scenario, but they can be considered a special type of the Transport Mobility Delay Models (TTMDM) below). In this class, the mobility of objects is more structured and less stochastic than those in the pedestrian models because they are usually constrained by fixed infrastructure (e.g. roadways, train tracks etc.). However, based on the infrastructure, we can easily make a distinction between two naturally different types of vehicular mobility patterns, we call the first type *traffic-based transport* and the second *non-traffic-based transport*. Traffic-based Transport Mobility Delay Model (TTMDM) is concerned with objects whose mobility is governed by uncertain parameters that, in some cases, could affect the mobility behaviour significantly; this model describes the mobility of objects like buses, coaches and similar road-based public transports whose mobility pattern highly depends on the conditions of the road traffic, which cannot be precisely modelled/predicted in most cases. Non-traffic-based Transport Mobility Delay Model (NTMDM), on the other hand, is easier to develop because random delay factors (such as those in TTMDM) are of no or negligible effect on the entities' mobility behaviour. NTMDM is used to model the mobility of infrastructure-based public transports such as trains and underground tubes, where, apart from rare occasional signal and other minor failures, have deterministic mobility patterns.

**Traffic-based Transport Mobility Delay Model (TTMDM).** Most realistic traffic-based mobility models [53] adjust the velocity of objects in such a way to avoid collisions. However, this level of microscopic modelling is not required in our scenario because we are only concerned about the time it would take those objects to move from one point to another, not the actual movements they should make. Thus, we consider modelling the factors that will affect this time figure, which can be estimated

as follows:

$$t_{a,b} = \frac{d_{a,b}}{\bar{v}_{a,b}} + D_{a,b}^{\text{traffic}} + \sum D_{a,b}^{\text{interest}} + \sum D_{a,b}^{\text{abnormal}} \quad (7.2)$$

where  $d_{a,b}$  and  $\bar{v}_{a,b}$  are the distance and the maximum allowable speed of the roadway between points  $a$  and  $b$ , respectively,  $D_{a,b}^{\text{traffic}}$  is the expected traffic delay of the roadway between points  $a$  and  $b$  and can be estimated by the geographical and physical characteristics of the area as well as the current time of the day,  $D_{a,b}^{\text{interest}}$  are delays incurred by points of interest (POI) located between  $a$  and  $b$ , and finally  $D_{a,b}^{\text{abnormal}}$  represents abnormal events that reportedly occurred in the road segment between  $a$  and  $b$ , such as accidents, breakdowns etc., both  $D_{a,b}^{\text{interest}}$  and  $D_{a,b}^{\text{abnormal}}$  can be obtained offline either from public resources (e.g. maps) or from the police.

**Non-traffic-based Transport Mobility Delay Model (NTMDM).** Modelling non-traffic-based transport is clearly more straightforward because the uncertainty of the stochastic delays the traffic-based transports suffer from is largely eliminated (or mitigated) here. This class describes the mobility of vehicular entities with fixed infrastructure, such as trains, underground tubes etc. In this case, we can calculate the time an object takes from point  $a$  to point  $b$  as follows:

$$t_{a,b} = \frac{d_{a,b}}{v_{a,b}} + D_{a,b}^{\text{stoppage}} + D_{a,b}^{\text{deceleration}} + \sum D_{a,b}^{\text{abnormal}} \quad (7.3)$$

where  $v_{a,b}$  is the fixed speed of the object on its journey from  $a$  to  $b$  spanning a distance  $d_{a,b}$  (which can be obtained offline),  $D_{a,b}^{\text{stoppage}}$  is the (almost) fixed stoppage time at stations,  $D_{a,b}^{\text{deceleration}}$  models the deceleration behaviour around the stations (in equation 7.2, the deceleration effect is part of the  $D_{a,b}^{\text{traffic}}$  variable), and  $D_{a,b}^{\text{abnormal}}$  represents any event that may cause a delay to the service, such as signal failure.

### 7.4.3 Multi-modal Mobility Delay Model

Conventionally, when modelling the mobility of a particular object, we implicitly assume that the object's behaviour will be consistent throughout the simulation. However, in our scenario we cannot rule out the possibility that the target could have used multiple different transportation modes, each with different mobility characteristics. Thus far, we have introduced three mobility delay models (PMDM, TTMDM, NTMDM) and now we model the transition between them by constructing a Multimodal

Model	PMDM	TTMDM	NTMDM
PMDM	Trivial	Level-1	Level-1
TTMDM	Trivial	Level-2	Level-2
NTMDM	Trivial	Level-2	Level-2

Table 7.1: Transition modelling between PMDM, TTMDM and NTMDM models

Mobility Delay Model (MMDM)<sup>4</sup> to assure a continuous flow of the target. Essentially, MMDM will only model the *transition* behaviour between two different models (or two different carriers of the same model) because once the transition is completed, the relevant mobility model is called to simulate the mobility of the next part of the journey until another transition is required. The delay incurred by this transition is called *Inter-Transport Delay* (ITD) as briefly discussed in section 3.4.3. When the transition happens between two carriers of the same mobility model (e.g. changing bus), the transition is said to be *homogenous*, otherwise if the transition happens between two carriers of different mobility models (e.g. transition from bus to train), the transition is said to be *heterogeneous*. Table 7.1 lists all possible transition combinations along with the kind of transition in each case (these transition actions are described below).

In a vehicular setting, we are actually interested in tracking the individual who is being transported by a carrier vehicle, not the vehicle itself, thus it is the entity who makes the transition which we need to model. Clearly, in PMDM both the entity and the carrier are a single component. When an entity shifts from any model to PMDM, the transition is smooth and incurs no delay (i.e., an individual does not have to wait before commencing a *walk* action). For any other situation, however, transition modelling is required to calculate the time an entity will need to wait before shifting to the next model (or carrier). The main idea is to observe the timetables of the carriers at the transition location and calculate the transient wait time. Table 7.2 is an excerpt from a real bus timetable and is organised by stops (represented by rows) and journeys (represented by columns). As shown in table 7.2, a particular object (individual) rides in a journey that makes several stops, where the timetable indicates when that object will arrive at each subsequent stop; thus, while a particular model (e.g., TTMDM) will describe the mobility patters of several carriers (e.g., buses), there is a dedicated timetable for every carrier.

For level-1 transition, the entity is shifting from PMDM to either TTMDM or

<sup>4</sup>It appears that most research in multi-modal mobility modelling was done in robotic motion planning, e.g., [25]. To the best of our knowledge, apart from integrated modules in, e.g., VISSIM [106] and SUMO [78] simulators, no pervious formal multi-modal mobility modelling was proposed for tracking applications.

Stop 1	0328	0428	0528	0558	0618	0651	0713	0734	0810	0848	0918	0948
Stop 2	0331	0431	0531	0601	0621	0654	0716	0737	0813	0851	0920	0951
Stop 3	0336	0436	0536	0606	0626	0659	0721	0742	0819	0857	0925	0956
Stop 4	0342	0442	0542	0612	0632	0705	0727	0748	0825	0903	0931	1002
Stop 5	0349	0449	0549	0619	0639	0711	0735	0755	0831	0909	0938	1009
Stop 6	0357	0457	0557	0627	0649	0721	0745	0805	0841	0916	0945	1016

Table 7.2: Sample bus timetable (excerpt from real bus timetable)

NTMDM, in both cases, the entity will most likely experience a slight transient delay due to the time difference of when it arrives at location  $x$  and when the next carrier belonging to the intended model stops at it. In this case, as soon as the entity arrives in  $x$ , it checks the intended carrier's timetable for the next departure time at its current location based on the current time and calculates the time difference. Level-2 transition is similar except it models the transition between TTMDM and NTMDM. We now describe this transition process in details.

Recall that in our scene graph  $G_M$  we represent roadways by edges  $\mathcal{E}^{G_M}$  and intersections by vertices  $\mathcal{V}^{G_M}$ . The transition can only happen in an intersection, so let  $v_i \in \mathcal{V}^{G_M}$  be a transition vertex which  $n$  carriers from either TTMDM or NTMDM stop at, let these carriers be denoted by  $R_1, R_2, \dots, R_n$  (this information is included in the label of  $v_i$ , see section 7.3). The first step is to obtain the timetables of these  $n$  carriers  $T^{R_1}, T^{R_2}, \dots, T^{R_n}$ , and covert them into matrices  $M^{R_1}, M^{R_2}, \dots, M^{R_n}$ , where the rows represent stops and the columns represent journeys.

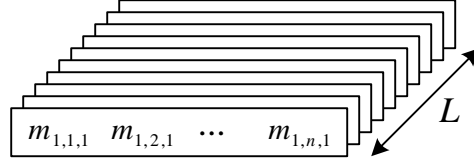
$$\begin{pmatrix} m_{1,1}^{R_1} & m_{1,2}^{R_1} & \cdots & m_{1,e}^{R_1} \\ m_{2,1}^{R_1} & m_{2,2}^{R_1} & \cdots & m_{2,e}^{R_1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{c,1}^{R_1} & m_{c,2}^{R_1} & \cdots & m_{c,e}^{R_1} \end{pmatrix} \cdots \begin{pmatrix} m_{1,1}^{R_n} & m_{1,2}^{R_n} & \cdots & m_{1,l}^{R_n} \\ m_{2,1}^{R_n} & m_{2,2}^{R_n} & \cdots & m_{2,l}^{R_n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{k,1}^{R_n} & m_{k,2}^{R_n} & \cdots & m_{k,l}^{R_n} \end{pmatrix}$$

Note that the dimensions of the matrices depend on the timetables and may be different for different carriers. Next, we extract the rows corresponding to  $v_i$  from  $M^{R_1}, M^{R_2}, \dots, M^{R_n}$  and create a 3D matrix  $M^{v_i}$  by superimposing these rows; this is illustrated in figure 7.5.

The dimensions of this new 3D matrix  $M^{v_i}$  will be  $1 \times n \times L$ , such that

$$L = \max\{w(M^{R_1}), w(M^{R_2}), \dots, w(M^{R_n})\}$$

where  $w(M)$  is the width (number of columns) of matrix  $M$ . In other words,  $L$  is

Figure 7.5: Illustration of the 3D matrix  $M^{v_i}$ 

the number of journeys that are being made by the carrier  $R_i$  that makes the highest number of journeys, where  $i \in \{1, 2, \dots, n\}$ . Obviously, if  $R_1, R_2, \dots, R_n$  do not all make the same number of journeys,  $M^{v_i}$  will contain some undefined values (set to 0). We assume access to a global clock which upon calling the procedure  $CurrentTime()$ , it returns the current time. Once  $M^{v_i}$  is created,  $c = CurrentTime()$  is obtained to build a  $1 \times n$  matrix  $\hat{M}^{v_i} = [m_{1,1}, m_{1,2}, \dots, m_{1,n}]$  such that:

$$m_{1,j} = \begin{cases} |c - m_{1,j,z}| + \epsilon & \text{if } z \geq c \geq z + 1, \\ \epsilon & \text{if } c = z \text{ or } c = z + 1, \\ \infty & \text{otherwise.} \end{cases}$$

where  $z \in \{1, 2, \dots, L\}$ , and  $\epsilon$  is a random delay representing the various factors that may hold the carriers off (examples of such factors are discussed in section 6.5), plus the wait time at each stop. The matrix  $\hat{M}^{v_i}$  will now indicate how long an entity at the current location  $x$  has to wait to pick any carrier  $R_1, R_2, \dots, R_n$  passing by  $x$  (regardless of whether  $R_1, R_2, \dots, R_n$  belong to the same or different model); the matrix will list all the carriers stopping at  $v_i$  along with the estimated delay figures for each.

## 7.5 Fuzzy Trace Validation

We assume (realistically) that the only available traces of the target came from recordings made by CCTV cameras. However, it is likely that some CCTV images will slightly be defected (e.g., blurred etc.) or basically taken for overcrowded areas where it is difficult to completely ascertain the presence of the target. Thus, some of the target's available traces may have not been generated reliably and need to be validated before proceeding to the actual trace reconstruction process. Let  $K^s = \{k_1^s, k_2^s, \dots, k_m^s\}$  be a set containing all the available (raw) traces for a target  $s$ . Each element (trace) in  $K^s$  is associated with 4 pieces of information  $k_i^s = (l_i^s, t_i^s, q_i^s, C_{ID})$ , representing the location  $l_i^s$  and the time  $t_i^s$  at which the target  $s$  was observed by the CCTV camera  $C_{ID}$ , and  $q_i^s$  is a measure of the quality of the CCTV image of which  $k_i^s$  was generated from (for

convenient, we'll often drop the  $C_{ID}$  parameter). In practice,  $K^s = N^s \cup U^s$ , where  $N^s$  and  $U^s$  are the sets of reliable and unreliable records of the target  $s$ , respectively. Usually, the reliability of  $k_i^s$  is determined by  $q_i^s$  where the target is thought to have been observed but due to a low quality CCTV image, this cannot be confirmed reliably. Thus, before proceeding to the trace reconstruction phase, we first need to evaluate elements in  $U^s$  and then decide whether to accept or reject them.

First, we carry out an initial test where we assume we can calculate the shortest path between any two points in the tracking graph using, e.g., Dijkstra algorithm [33]. Suppose we have  $K^s = \{k_{n-1}^s, k_n^s, k_{n+1}^s\}$  and that  $k_n^s \in U^s$  while  $k_{n-1}^s, k_{n+1}^s \in N^s$ . If the following is satisfied

$$t_{n+1}^s - t_n^s < SP^{time}(l_n^s, l_{n+1}^s)$$

then we reject  $k_n^s$  immediately, where  $SP^{time}(A, B)$  returns the time it takes an entity to travel from  $A$  to  $B$  using the shortest path between  $A$  and  $B$ . That is, there is no way the target could have moved from  $l_n^s$  to  $l_{n+1}^s$  in less than it would have taken him using the shortest route between these two points. Similarly, we also reject  $k_n^s$  if the following is satisfied:

$$t_n^s - t_{n-1}^s < SP^{time}(l_{n-1}^s, l_n^s)$$

### 7.5.1 Fuzzy Logic

If the initial test succeeds (if  $k_i^s$  is not rejected), we proceed by building a fuzzy inference system (FIS) to evaluate the trustworthiness of the records in  $U^s$ . Generally, fuzzy logic [134] is used to model imprecise concepts, such as temperature, height etc. Unlike the classical crisp sets where an element is either inside or outside the set, fuzzy sets do not have rigid boundaries where elements may belong to more than one fuzzy set to certain degrees. Thus, every fuzzy set  $A$  has a membership function  $\mu_A$  which decides to what extent elements belong to  $A$ . Figure 7.6 illustrates the three main phases (components) of a FIS. In the first phase, the input is *fuzzified*, where several fuzzy sets are created and their corresponding membership functions convert the crisp input into fuzzy one. Then, a set of fuzzy IF-THEN rules are applied to the fuzzy input, and finally the result is *defuzzified* to return a crisp output. In our case, we try to evaluate how much we should trust the traces of  $U^s$ . Let  $U^s = \{u_1^s, u_2^s, \dots, u_p^s\}$ , and  $p < |N^s|$ . Since each trace  $u_i^s$  represents 3 variables ( $l_i^s, t_i^s, q_i^s$ ), we use these as inputs to our fuzzy system and evaluate each  $u_i^s$  independently.



Figure 7.6: Fuzzy Inference System

### 7.5.2 Fuzzification

Based on the input (the  $u_i^s$  records), we have three fuzzy variables,  $L = \{l_1^s, l_2^s, \dots, l_p^s\}$ , representing location,  $T = \{t_1^s, t_2^s, \dots, t_p^s\}$ , representing time, and  $Q = \{q_1^s, q_2^s, \dots, q_p^s\}$ , representing the quality of the CCTV images, where  $|U^s| = p$ . Each variable is associated with a number of fuzzy sets as follows:  $L \rightarrow \{POI, N\}$ , where POI represents a Point of Interest (e.g. a shopping mall), and  $N$  represents non-POI areas,  $T \rightarrow \{D_1, P_1, D_2, P_2, D_3\}$ , where  $D_1$  is morning,  $D_2$  is afternoon,  $D_3$  is night, and  $P_1, P_2$  are the morning and afternoon peak periods<sup>5</sup>,  $Q \rightarrow \{G_1, G_2, G_3\}$ , where  $G_1$  is good,  $G_2$  is average and  $G_3$  poor. Every set belonging to every variable has its own membership function  $\mu_x$ , where

$$x \in \{POI, N, D_1, P_1, D_2, P_2, D_3, G_1, G_2, G_3\}$$

All membership functions are normally distributed. Figure 7.7 shows plots of the three input variables  $Q, L, T$  and the output variable  $O$ . These membership functions were created based on subjective analysis to capture the characteristics of the variables they model.

### 7.5.3 Fuzzy Inference

Our fuzzy inference formulation is developed by devising a set of IF-THEN rules that implement our fuzzy reasoning. These rules accept inputs from the  $(L, T, Q)$  variables and return a value assigned to the output variable  $O$ , which in turn has two fuzzy sets,  $Y, X$ , to indicate how much trustworthy the record  $u_i^s$  is, where  $Y$  and  $X$  represent trustworthiness and untrustworthiness, respectively. The IF-THEN rules are based on the conditional operators AND and OR, but these operators cannot be directly applied on fuzzy values. The most popular fuzzy counterpart is to replace the AND and OR operators by the *minimum* and *maximum* functions, respectively; that is, the expressions  $(A = a) \wedge (B = b)$  and  $(A = a) \vee (B = b)$  are translated to  $\min\{\mu_A(a), \mu_B(b)\}$  and  $\max\{\mu_A(a), \mu_B(b)\}$ , respectively. Table 7.3 summarises our fuzzy rules. Basically, these rules evaluate the trustworthiness of a record  $u_i^s$  given when, where and how it

<sup>5</sup>This day classification assumes a week day. With a few adjustments to the IF-THEN rules in the fuzzy inference phase, this can easily be modified for weekends where there are no peak periods.

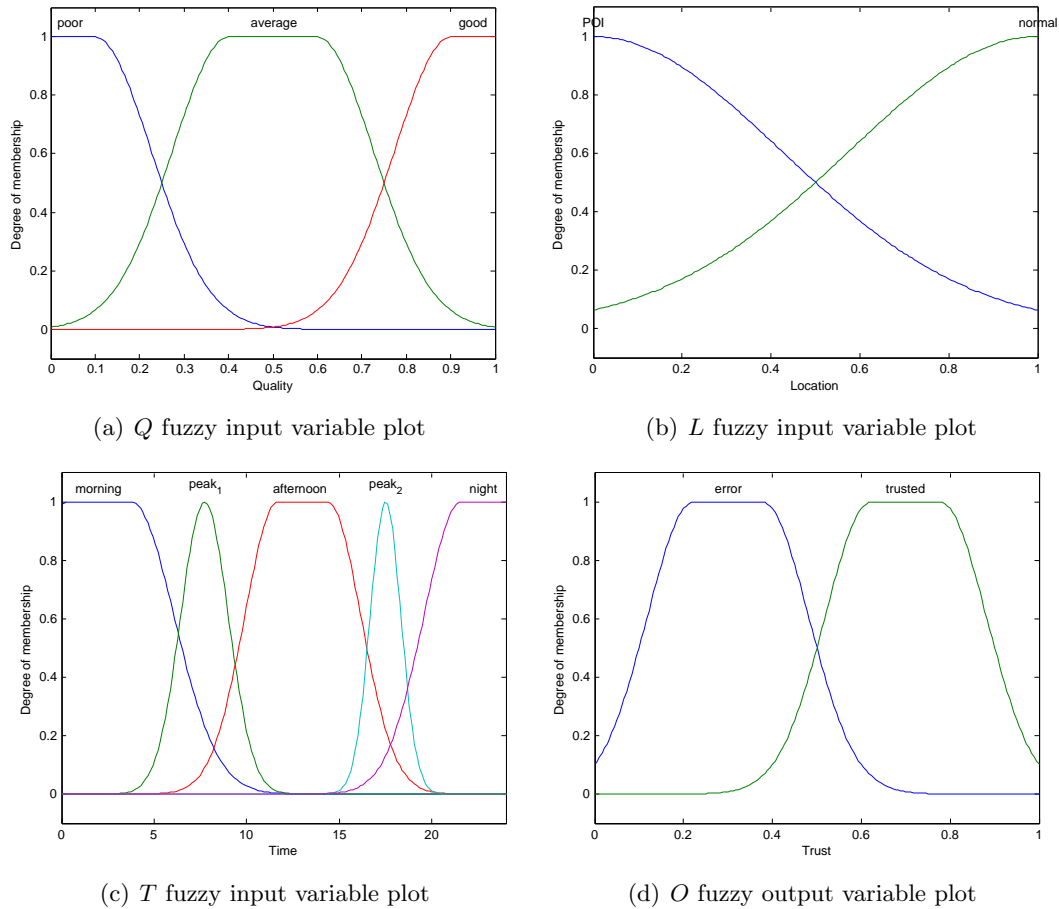


Figure 7.7: Input and output variables plot

was recorded. Intuitively, the more crowded the recording location is, the less confident we are that the target was spotted correctly; this also applies to low quality CCTV footage and records made during peak hours.

#### 7.5.4 Defuzzification

The fuzzy rules in table 7.3 will return a fuzzy value belonging to the fuzzy output variable  $O$  where the membership functions of  $O$ 's two fuzzy sets,  $Y$  and  $X$ , will assign a degree of membership to each set. However, a fuzzy output is not very meaningful in this case, so we need to defuzzify it and return a crisp value that indicates whether we should accept or reject the corresponding  $u_i^s$  record. There are different defuzzification methods [111], we adopt the centroid method which is the most popular. In this method, the centre of the curve, formed by evaluating the IF-THEN rules, is returned. Figures 7.8(a) and 7.8(b) show how our fuzzy system behaves for different values of

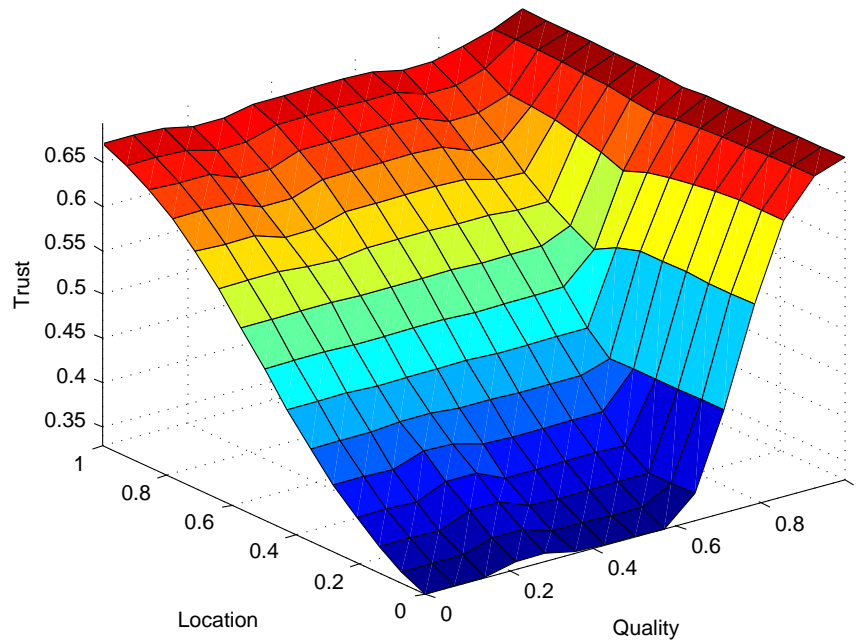
IF	THEN
$(Q = G_1) \wedge \{(T = D_1) \vee (T = D_3)\}$	$O = Y$
$(Q = G_1) \wedge (L = N) \wedge \{(T = P_1) \vee (T = P_2)\}$	$O = Y$
$(Q = G_1) \wedge (T = D_2) \wedge \{(L = POI) \vee (L = N)\}$	$O = Y$
$(Q = G_1) \wedge (L = POI) \wedge \{(T = P_1) \vee (T = P_2)\}$	$O = X$
$(Q = G_2) \wedge \{(T = D_1) \vee (T = D_3)\}$	$O = Y$
$(Q = G_2) \wedge (L = N) \wedge (T = D_2)$	$O = Y$
$(Q = G_2) \wedge (L = N) \wedge (T = P_1)$	$O = Y$
$(Q = G_2) \wedge (L = N) \wedge (T = P_2)$	$O = Y$
$(Q = G_2) \wedge (L = POI) \wedge (T = P_1)$	$O = X$
$(Q = G_2) \wedge (L = POI) \wedge (T = P_2)$	$O = X$
$(Q = G_2) \wedge (L = POI) \wedge (T = D_2)$	$O = X$
$(Q = G_3) \wedge \{(T = D_1) \vee (T = D_3)\}$	$O = Y$
$(Q = G_3) \wedge \{(T = P_1) \vee (T = P_2)\}$	$O = X$
$(Q = G_3) \wedge (T = D_2) \wedge (L = N)$	$O = Y$
$(Q = G_3) \wedge (T = D_2) \wedge (L = POI)$	$O = X$

Table 7.3: Fuzzy rules

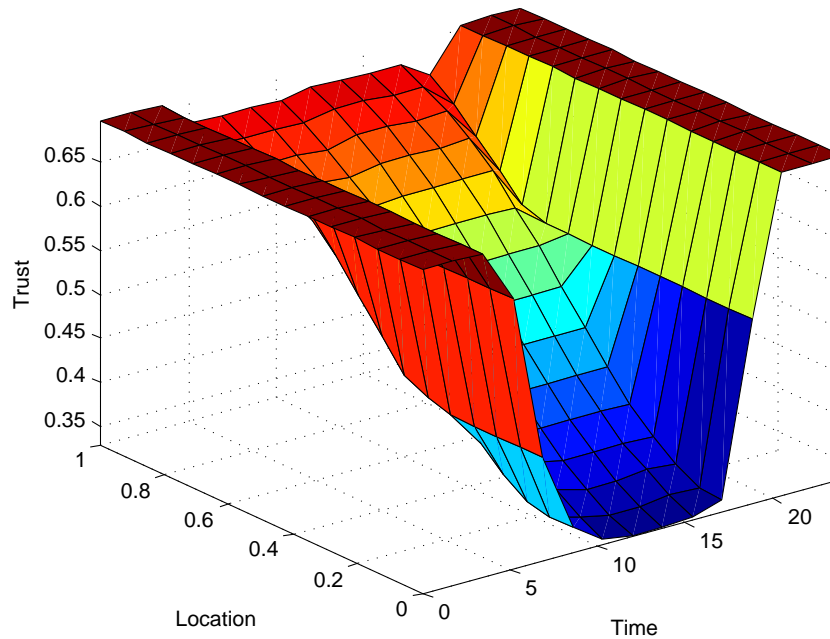
the three input variables ( $L$ ,  $T$ ,  $Q$ ). In figure 7.8(a), both the location  $L$  and quality  $Q$  variables are plotted over a range of  $[0,1]$ . As we approach 0, the quality of the CCTV gets worse and the location variable tends to approach a POI area. Clearly, the closer we are to a POI and the lower the image quality is, the less trust we have on the corresponding record. Similarly, the time variable  $T$  in figure 7.8(b) is plotted over a range of  $[0,24]$ , representing the daily 24 hours. As the figure shows, we have less trust in records taken around POI areas during daytime, but for the early hours of morning and late hours of night, we trust most of the records regardless of the location as it is likely that during these periods most locations are not crowded enough to defect the CCTV images (note that here we ignore the day light effect).

## 7.6 Trace Reconstruction

As briefly mentioned in section 6.1, classical missing data algorithms, such as EM [32] and data augmentation [124], cannot be directly used to reconstruct traces because these algorithms mainly make statistical inferences based on incomplete data, but will not reconstruct it, as required in our case. Additionally, we cannot assume that we



(a) Location-quality plot



(b) Location-time plot

Figure 7.8: Graphical representation of the relationship between the input/output variables of our fuzzy system

have a sufficiently large number of available traces to be able to use these algorithms. Iterative sampling algorithms, such as Markov Chain Monte Carlo (MCMC) [47], when

adapted for a missing data setting, cannot be used here too for the same reasons. Instead, we take a different algorithmic approach to fill the *gaps* formed by the missing traces. We develop an efficient reconstruction algorithm that, using mobility delay models (as introduced in section 7.4), selects the route(s) that the target has most likely taken through a gap, given the time it spent traversing it. In the worst case, the algorithm would at least eliminate several routes that the target could not possibly have taken, which may still provide important evidence.

### 7.6.1 The Algorithm

The reconstruction algorithm  $\mathcal{A}_R$  first considers each gap individually, reconstructs it, repeats for all gaps, and then connects the reconstructed gaps to obtain the full trace of a target  $s$ . Abstractly,  $\mathcal{A}_R$  consists of two fundamental building blocks, (1) a multi-graph traversing algorithm called Weight-Bound-Search (WBS), and (2) several mobility models. Once  $\mathcal{A}_R$  is executed, it proceeds by running WBS over a gap, WBS, in turn, repeatedly calls the mobility models (possibly via  $\mathcal{A}_R$ ) and returns a route (or routes) connecting the gap;  $\mathcal{A}_R$  then reconstructs the other gaps in a similar fashion. The WBS algorithm is based on a branch-and-bound approach [81] to optimise the reconstruction process, and uses a *crawler* that traverses the gaps and finds plausible routes. For a gap  $G_p : v_m \rightarrow v_n$  between vertices  $v_m$  and  $v_n$ , where  $m, n \in \{1, 2, \dots, |\mathcal{V}^{GM}|\}$ , a crawler  $C^{G_p}$  is generated at  $v_m$  and broadcasted toward  $v_n$ . The crawler  $C^{G_p}$  maintains two data structures: (1) a LIFO<sup>6</sup> list of vertices and edges traversed so far  $\chi_{C^{G_p}}$ , and (2) a delay variable  $\tau_{C^{G_p}}$ . The  $\chi_{C^{G_p}}$  is dynamically updated whenever  $C^{G_p}$  traverses a vertex or edge to keep track of all the vertices and edges the crawler  $C^{G_p}$  has visited. The delay variable  $\tau_{C^{G_p}}$  is initially set to 0 and is also dynamically updated whenever  $C^{G_p}$  traverses an edge or an  $S$ -vertex (but not  $U$ -vertex, see below). When  $C^{G_p}$  is first initiated at  $v_m$ , it checks  $v_m$ 's label  $\ell_m = \{R_x^y, \dots, R_k^l\}$  which contains information about the routes that  $v_m$  is part of and consequently finds the set of its next-hop neighbouring vertices  $\hat{\ell}_m$ , where

$$\hat{\ell}_m = \begin{cases} |\ell_m| & \text{if } v_m \notin \overleftarrow{\rho}^G, \\ |\ell_m| - k & \text{if } v_m \in \overleftarrow{\rho}^G. \end{cases}$$

and  $k$  is the number of times  $v_m$  appears in  $\overleftarrow{\rho}^G$  (the number of routes in which  $v_m$  is a tail-end vertex; such routes terminate at  $v_m$  and thus do not have next-hop). However, since we are considering a multi-graph, it is possible that some of these

<sup>6</sup>Last-In-First-Out is a stack data structure, which organises elements in descending order such that the last element entering the queue is the first element leaving it.

routes are passing by the same next-hop neighbour (creating parallel edges between two vertices), so the  $\hat{\ell}_m$  list may actually contain repeated vertices. Even if this is the case, we still need to consider each outgoing edge (even if all edges are parallel) separately because it may have different weight depending on which route it belongs to. Thus, once all next-hop neighbours are found,  $C^{G_p}$  selects one of them, say  $v_u$ , finds the edges (routes) between  $v_m$  and  $v_u$ , that is  $\{e_i | e_i : v_m \rightarrow v_u\}$ , and selects one  $e_i$ . Once an  $e_i$  is selected,  $C^{G_p}$  tags it as “visited”, updates  $\chi_{C^{G_p}}$  and traverses it. It is important that  $C^{G_p}$  tags any edge it traverses as “visited” so it does not revisit it again and enters in an infinite loop. Furthermore, if  $C^{G_p}$  arrives at a vertex  $v_k$  and found that there is only one unvisited edge  $e_i$ , it tags  $e_i$  as “visited”, traverses it and then tags  $v_k$  as “exhausted” so it skips  $v_k$  if  $v_k$  ever happened to be a neighbour to some vertex  $C^{G_p}$  traverses in the future. Based on the type of the edges connecting  $v_m$  with its next-hop neighbouring vertices,  $C^{G_p}$  calls the appropriate mobility model (either PMDM, TTMDM or NTMDM) to calculate the delay of that edge, and updates its  $\tau_{C^{G_p}}$  as follows  $\tau_{C^{G_p}} = \tau_{C^{G_p}} + t_{v_x, v_y}$ , where  $t_{v_x, v_y}$  is the delay returned for the edge  $e_i : v_x \rightarrow v_y$  by the relevant mobility model (this applies to both  $R$ -edges and  $W$ -edges). Similarly, once  $C^{G_p}$  reaches an  $S$ -vertex  $v_y$ , it again updates  $\tau_{C^{G_p}}$  but this time by calling MMDM, such that  $\tau_{C^{G_p}} = \tau_{C^{G_p}} + t_{v_y}$  where  $t_{v_y}$  is the delay assigned to  $v_y$  by MMDM. However, since there is no transition between mobility models in  $U$ -vertices, MMDM is not called when reaching a  $U$ -vertex. The crawler traverses the various routes by repeatedly *backing-up* whenever its counter expires, being as a result of finding a plausible or implausible route. The back-up procedure proceeds as follows: once  $C^{G_p}$  finds an (im)plausible route, it checks its  $\chi_{C^{G_p}}$  and traverses backward through the edge in  $\chi_{C^{G_p}}[1]$  toward the vertex  $\chi_{C^{G_p}}[2]$ , where  $\chi[n]$  is the  $n$ -th element of the list  $\chi$ . It then deletes these two elements from  $\chi_{C^{G_p}}$ , and repeats the whole traversal process again (searching for neighbouring vertices etc.), but this time it does not traverse the edge it just came from because it is now tagged as “visited”. The crawler  $C^{G_p}$  backs-up if:

1.  $\tau_{C^{G_p}} + \epsilon > t_{v_n, v_m}$ , where  $\epsilon$  is a random value<sup>7</sup>, or
2. traversed a vertex/edge that already exists in its  $\chi_{C^{G_p}}$ , or
3.  $v_n$  (the other end of the gap) is reached, or
4. it reaches a vertex  $v_j$  such that  $v_j$  is a tail-end in all its routes (i.e.,  $v_j$  is childless).

In (1), the crawler backs-up once its  $\tau_{C^{G_p}}$  reaches a value greater than  $t_{v_n, v_m}$  (the time difference between when the target was observed at  $v_m$  and later at  $v_n$ —the two

<sup>7</sup>The value of  $t_{v_n, v_m}$  can be adjusted in case the crawlers failed to return any route.

ends of a gap  $G_p$ ), and  $\epsilon$  is a constant delay. This means that the target would take much longer than  $t_{v_m, v_n}$  if it had traversed that route, which is not possible. In (2), we only accept loop-free routes because this is what a rational target will opt to do (and also to prevent infinite loops), so if  $C^{G_p}$  reaches a vertex  $v_i$  such that  $v_i \in \chi_{C^{G_p}}$ , then it backs-up. We can also detect these vertices by checking whether they are tagged as “visited”, which implies that they are in the  $\chi_{C^{G_p}}$  set. In (3), once  $C^{G_p}$  reaches  $v_n$ , it checks its  $\tau_{C^{G_p}}$ , if  $\tau_{C^{G_p}} + \epsilon \leq t_{v_m, v_n} - \epsilon$ , it backs-up (in other words, if a crawler returned a time much shorter than  $t_{v_m, v_n}$ , it is probably not the route the target has taken). Otherwise, if  $t_{v_m, v_n} - \epsilon \leq \tau_{C^{G_p}} \leq t_{v_m, v_n} + \epsilon$ , it backs-up, returns the route in  $\chi_{C^{G_p}}$  as a possible route the target may have taken, as well as returning the corresponding  $\tau_{C^{G_p}}$ . Finally, in (4)  $C^{G_p}$  also backs-up when it reaches a *childless* vertex  $v_j$ ; additionally, it tags  $v_j$  as “exhausted”. The WBS algorithm terminates when its crawler terminates and that happens when the crawler reaches a vertex in which all neighbouring (next-hop) vertices are tagged as “exhausted”, this means that they have been already extensively traversed (i.e., all their outgoing edges are tagged as “visited”). Algorithm 7.8 provides a pseudocode for the WBS algorithm, which is the heart of the reconstruction algorithm  $\mathcal{A}_R$ . Algorithm 7.8 accepts the scene graph with well marked gaps as well as the weight of each gap (the delay through the gap). Variables in the algorithm that are not explicitly initialised, are set to 0, while Boolean variables are initialised to false.

**Proposition 7.6.1.** *Given a finite search graph, the Weight-Bound-Search (WBS) algorithm will eventually terminate, with or without returning valid routes.*

*Proof.* Since the WBS is a weight-based algorithm, it is guaranteed to stop traversing a particular route  $R_i$  whenever its weight counter  $\tau_{C^{G_p}}$  expires (i.e.,  $\tau_{C^{G_p}} \geq t_{v_m, v_n} + \epsilon$ , where  $t_{v_m, v_n}$  is the delay through gap  $G_p : v_m \rightarrow v_n$ , and  $\epsilon$  is a small constant). Thus the only way for the algorithm to run indefinitely is if it gets into an infinite loop, which can only happen when it traverses the same route and backs-up over and over again. However, a route  $R_i$  cannot be traversed more than once because the algorithm tags every visited edge and would not traverse any tagged edge, so as long as there is finite number of edges in a graph, the algorithm will terminate.  $\square$

In addition, the WBS algorithm will also terminate when traversing an infinitely *deep* graph because it traverses the graph down to the point when its weight counter  $\tau_{C^{G_p}}$  expires. However, the WBS algorithm may fail to terminate when it runs over an infinitely *wide* graph (the node of the current level has infinitely many children). This, nevertheless, contributes to the completeness of the WBS algorithm.

**Algorithm 7.8** Weight-Bound-Search (WBS)

---

```

1: WBS(Graph,Source,Destination,GapWeight)
2: getNeighbours(Source)  $\rightarrow$  nei {list of neighbours}
3: for  $i = 1$  to  $i \leq \text{sizeof}(\text{nei})$  do
4:   if nei[i] = “exhausted” then
5:     Remove(nei[i]) {remove nei[i] if it is tagged as “exhausted”}
6:     Continue {loop to the next neighbour nei[i + 1]}
7:   end if
8:   getEdges(nei[i])  $\rightarrow$  nei[i].edges {get all nei[i]’s outgoing edges}
9:   for  $j = 1$  to  $j \leq \text{sizeof}(\text{nei}[i].\text{edges})$  do
10:    if nei[i].edges[j]  $\neq$  “visited” then
11:      Counter++ {How many “unvisited” edges nei[i] has}
12:      nei[i].edges[j]  $\rightarrow$  nei[i].currentEdge
13:      nei[i].edges[j] = “visited” {tag as “visited”}
14:      nei[i].untraversedEdge = true
15:    end if
16:  end for
17:  if nei[i].untraversedEdge = true then
18:    if Counter = 1 then {if there was only 1 “unvisited” edge}
19:      nei[i] = “exhausted” {tag nei[i] as “exhausted”}
20:    end if
21:    break {break out of the if statement and traverse nei[i + 1]}
22:  end if
23: end for
24: if nei =  $\perp$  then
25:   Terminate {terminate WBS if all neighbours are “exhausted”}
26: end if
27: for  $i = 1$  to  $i \leq \text{sizeof}(\text{nei})$  do
28:   getWeight(nei[i].currentEdge) = wei[i] {call mobility model to assign a weight}
29:    $\tau = \tau + \text{wei}[i]$  {update the time variable counter}
30:   pushRoute(nei[i].currentEdge, nei[i])  $\rightarrow$   $\chi$  {update route list}
31:   if nei[i] = Destination then
32:     if GapWeight -  $\epsilon \leq \tau \leq$  GapWeight +  $\epsilon$  then
33:       return  $\tau, \chi$ 
34:       Back-up(nei[i].currentEdge, nei[i], wei[i],  $\tau, \chi$ ) {back 1 level up to Source}
35:       Continue {Continue iterating the next neighbour}
36:     else if  $\tau \geq$  GapWeight +  $\epsilon$  then
37:       Back-up(nei[i].currentEdge, wei[i],  $\tau, \chi$ ); Continue
38:     end if
39:   else if nei[i]  $\in \overleftarrow{\rho}^G$  or nei[i]  $\in \chi$  then {if childless/End-tail vertex is reached}
40:     Back-up(nei[i].currentEdge, wei[i],  $\tau, \chi$ ); Continue
41:   else if  $\tau \leq$  GapWeight -  $\epsilon$  then
42:     WBS(Graph,nei[i],Destination,GapWeight) {recursion}
43:   end if
44: end for

```

---

**Procedure 7.9** Back-up Procedure

- 
- 1: **Back-up**(edge, vertex, weight,  $\tau$ ,  $\chi$ )
  - 2: Remove(edge, vertex,  $\chi$ ) {remove “edge” and “vertex” from  $\chi$ }
  - 3:  $\tau = \tau - \text{weight}$  {update  $\tau$  by subtracting “weight”}
  - 4: **return**  $\tau, \chi$
- 

**Proposition 7.6.2.** *Given a finite search graph, the Weight-Bound-Search (WBS) algorithm is complete. If there exist one or more solutions, WBS will return them all.*

*Proof.* For a gap  $G_p : v_m \rightarrow v_n$ , a valid solution means that there is a route  $R_i : v_m \rightarrow v_n$  with a weight  $\tau$  such that  $t_{v_m, v_n} - \epsilon \leq \tau \leq t_{v_m, v_n} + \epsilon$ . The fact that the crawler  $C^{G_p}$  will eventually terminate, as shown in proposition 7.6.1, means that it will traverse all valid and invalid routes and will terminate only when there are no more edges to traverse. Therefore, it follows that if there is a  $R_i$  that is a solution as outlined above, the crawler  $C^{G_p}$  will find it. Hence, the algorithm is complete.  $\square$

Once the crawler terminates, and there is more than one route returned, the algorithm selects the *best-fit* route, such that

$$|\chi_{C_f^{G_p}}| = \min\{|\chi_{C_1^{G_p}}|, |\chi_{C_2^{G_p}}|, \dots, |\chi_{C_n^{G_p}}|\}$$

That is, the route with less hops will be selected because this is what a rational target would probably do (choose a route that does not have many stops). Additionally, by observing the labels of the edges and vertices of the returned routes, a preferred route can be selected that minimises the number of transitions between different mobility models and/or carriers of the same model. However, even if no single preferred route was selected, the algorithm is likely to minimise the possible routes significantly.

### 7.6.2 Complexity Analysis

It is not clear how to derive accurate complexity bounds for the trace reconstruction algorithm  $\mathcal{A}_R$  without considering a particular scenario because the cost of reconstructing a gap depends on many factors that can only be determined on per scenario basis. It is also almost certain that gaps will not be of the same size and thus will have different complexities. Although the back-up and termination conditions were designed in such away that maximises the efficiency of the algorithm, in this section we consider the worst case scenario and provide estimates of the algorithm’s complexity bounds.

The complexity of  $\mathcal{A}_R$  is essentially the cost of running the WBS algorithm plus the cost of calling the mobility models when requested by WBS, which happens whenever WBS traverses an edge or vertex ( $S$ -vertex, to be precise). Clearly, the fewer the

traversed edges/vertices, the fewer the calls to the mobility models. This, in turn, depends on the size of the gap, and how many gaps need to be reconstructed before obtaining the full trace. Let  $\psi_P, \psi_T, \psi_N, \psi_M$  be the costs of running PMDM, TTMDM, NTMDM and MMDM, respectively, and let  $\psi_{WBS}$  be the cost of running WBS, then generally the computational complexity  $\psi_{\mathcal{A}_R}$  of the  $\mathcal{A}_R$  algorithm is:

$$\psi_{\mathcal{A}_R} = \mathcal{O}(\psi_P + \psi_T + \psi_N + \psi_M) + \mathcal{O}(\psi_{WBS})$$

The WBS algorithm is a heuristic<sup>8</sup> graph traversal algorithm [112]. When WBS runs over a tree, it is strictly equally or more efficient than the famous Breadth-First-Search (BFS) and Depth-First-Search (DFS) algorithms [29]. In the worst case scenario, both BFS and DFS have a time complexities of  $\mathcal{O}(b^d)$ , where  $b$  is a fixed branching factor (the number of the children of a vertex) and  $d$  is the depth of the graph (the number of descendant levels from the root node), and space complexity of  $\mathcal{O}(b^d)$  and  $\mathcal{O}(bd)$ , respectively. However, recall that beside  $b$  and  $d$ , WBS also involves a *weight factor*  $\tau$ , which determines when its crawler backs-up or terminates; that is, unlike BFS and DFS, WBS is not required to traverse all the vertices of the tree before it terminates (i.e., it backs-up once its weight  $\tau$  expires).

The hardness of deriving accurate complexity bounds for the WBS algorithm stems from the fact that we almost always run it over a mixed multigraph<sup>9</sup> (representing a road network) which is, unlike a tree, extremely unstructured.

**Definition 7.6.3** (Multigraph). *A multigraph is a graph that allows multiple edges between a pair of vertices (these are called parallel edges). A multigraph can be either directed, undirected or mixed (where some edges are directed and others are undirected).*

Observing our scene graph  $G_M$ , which is mostly based on modern, sufficiently mature, road networks, we conjecture that  $G_M$  is best represented by a multigraph with the following properties: (1) mixed, since it contains both  $R$ -edges and  $W$ -edges, (2) parallel edges are allowed, (3) loops are allowed, and (4) cycles are allowed (that is, the graph may contain cycles, where a path starts and ends at a particular vertex but traverses at least 1 additional vertex en-route; this is in contrary to loops, where a path

<sup>8</sup>While most heuristic algorithm, such as the A\* algorithm [99], tries to find the lowest-cost path, WBS aims at finding *all* plausible paths based on a particular weight, and these may not be the lowest-cost paths. Heuristic algorithms use extra information, e.g. weight, to improve the performance.

<sup>9</sup>Unfortunately, there has been a lot of speculation on what a multigraph actually is, while some authors explicitly require multigraphs to have parallel edges [104], others just permit them [49]—some authors, thus, consider graphs a special case of multigraphs in which there are no parallel edges. Similarly, another controversial issue is whether to allow [55] or disallow [52] loops (a.k.a. self-loop, an edge which starts and ends at the same vertex), where a multigraph with loops is sometimes called pseudograph [49]. To prevent ambiguity, we explicitly define what we mean by multigraph.

starts and ends at the same vertex without passing by any vertex en-route). We clearly allow parallel edges because most modern transport networks have multiple carriers spanning the same routes or at least have common routes during part of their journeys; these multiple routes are then represented by parallel edges. Although rare, loops do exist in some road networks, but this does not affect the WBS algorithm because these loops will be detected. Finally, obviously it is very common to have cycles in a road network (in fact, this is a common practice in road infrastructure planning).

To simplify the complexity analysis, we first prove the complexity bounds of WBS over a tree structure, then show that when running WBS over a multigraph it is generally equally or more efficient than running it over a tree.

**Proposition 7.6.4.** *In the worst case scenario, when running the Weight-Bound-Search (WBS) algorithm over a tree, it has a time complexity of  $\mathcal{O}(b^{\hat{\tau}})$  and space complexity of  $\mathcal{O}(\hat{\tau})$  where  $\hat{\tau} = \min\{\tau, d\}$ , and  $\tau$  is the weight assigned to the tree (the delay factor of the tree) while  $d$  is the depth of the tree.*

*Proof.* Let  $b$  be a branching factor and  $\tau$  be a weight factor, then in the worst case all edges have the same minimal weight, say 1, which will cause a crawler  $C^{G_p}$  to increase its temporary weight counter  $\tau_{C^{G_p}}$  every time it traverses an edge (or  $S$ -vertex) by 1 and eventually backs-up once the  $\tau_{C^{G_p}} > \tau + \epsilon$ . Similarly, if  $C^{G_p}$  reached the boundary of the graph (at depth  $d$ ) and  $\tau_{C^{G_p}} < \tau + \epsilon$  still holds,  $C^{G_p}$  will also back-up. Therefore, the time complexity of WBS is upper bounded by

$$1 + b + b^2 + \dots + b^{\hat{\tau}} = \mathcal{O}(b^{\hat{\tau}})$$

where  $\hat{\tau} = \min\{\tau, d\}$ . For the space complexity, WBS operates in a similar fashion as DFS traversing the graph as deep as  $\hat{\tau}$ , then backs-up, updates its route buffer (by removing the recently backed up edges/vertices) and repeat. However, at any give time, WBS is required to store only one route of length up to  $\hat{\tau}$ , which it dynamically updates throughout its execution.  $\square$

The introduction of  $\tau$  in WBS means that not all vertices of the graph will necessarily be traversed, but the worst case scenario is when  $\tau > d$  (i.e.,  $\tau$  is greater than the total weight of the most expensive route in a graph) in which case all vertices will be traversed—for a large graph, this is rarely the case though. Compared to BFS/DFS, in the worst case scenario the time complexity of WBS is similar to that of BFS/DFS, but WBS features a significant improvement on space complexity, since only one route needs to be stored at any given time while DFS and BFS have to store all the vertices and edges they traverse. Before proving that WBS traverses a multigraph equally or

more efficiently than a tree, we elaborate on how the branching factor  $b$  is represented in a multigraph. Conventionally,  $b$  is the number of the children of a particular node, and is fixed for the kind of trees we discussed so far (all nodes have the same  $b$ ), however since in a multigraph there may be parallel edges, and all these parallel edges need to be traversed,  $b$  should be equal to the number of the outgoing edges not the children vertices. Also, notice that in a multigraph,  $b$  is usually not fixed (there are vertices with more children – or more outgoing edges – than others).

**Proposition 7.6.5.** *Let  $b$  be the branching factor of two graphs  $G$  and  $M$ , both with depth  $d$ , and let  $G$  be a tree while  $M$  is a multi-graph. Then, running the Weight-Bound-Search (WBS) algorithm over  $M$  is equally or more efficient than running it over  $G$ .*

*Proof.* As shown in proposition 7.6.4, the complexity of WBS over a tree is  $\mathcal{O}(b^d)$ . However, since there is no fixed  $b$  in a multigraph scenario,  $b$  represents the number of edges going out off the vertex with the highest number of such edges (since this is an asymptotic bound). This means that  $b$  for other vertices in a multigraph is less than or equal to this asymptotic  $b$ . Hence  $complexity(M) = \bar{b}^d$  is less than or equal to  $complexity(G) = b^d$ , since  $b \leq \bar{b}$ .  $\square$

## 7.7 Summary

In this chapter, we conclude the contributions of the thesis by considering the largest (and most challenging) scenario so far, where we probabilistically reconstruct a target’s trace in a multi-modal environment (an environment combining both pedestrian and vehicular settings). We propose a trace reconstruction algorithm that, given information about several locations where the target was observed, it reconstructs its full trace by considering the various possible routes connecting the gaps formed by the periods where the target was unobservable. The algorithm first validates the available data (traces) by means of fuzzy logic, then uses a specially crafted mobility models to reconstruct a given trace. These models (called mobility delay models to distinguish them from the standard conventional mobility models) describe the time delays over a particular set of routes without delving into various microscopic mobility details of the individual nodes; such information is not important and will not necessarily improve the reconstruction process. Finally, we evaluated the complexity of the reconstruction algorithm and showed that the algorithm is both complete and optimal while comparing its complexity to the popular DFS (Depth First Search) and BFS (Breadth First Search) search algorithms.

## Chapter 8

# Conclusion and Future Work

Traditionally being a law enforcement application, digital forensics has recently found its way to academia and is today among the most active research areas. Digital forensics, however, is rapidly evolving and has recently given rise to the more intelligent discipline of computational forensics. In this thesis, we focused on a particular application of computational forensics, namely criminal surveillance and tracking. The main goal of this research is to investigate how to passively (and clandestinely) track a target, who we assume to be a suspect that may have been associated to either a crime that already took place, in which case we consider offline tracking (also known as trace reconstruction), or a crime that is expected to take place, in which case we consider online tracking. While in online tracking, the tracking process takes place in real-time, in offline tracking, the tracking process takes place after the fact (post hoc), where in the latter case any available trace associated to a target suspect is collected and forensically analysed. In online tracking, we emphasise that the tracking process should be passive such that the target is not aware of it. This mere requirement greatly contributes toward the novelty of our research since passive tracking has rarely been thoroughly investigated in the literature, and, evidently, is more difficult than active tracking as it adds the additional requirement of maintaining clandestinity throughout the whole tracking process. In contrast, when tracking a target actively, the target is aware of the tracking, sometimes is even cooperatively involved. While developing our online and offline tracking algorithms, we also considered different environments and scenario settings. Roughly, the thesis can be divided into three main parts:

- *Part 1: Pedestrian tracking.* In this part, we considered online tracking of individuals in real-time by observing signal emissions from any wireless devices they may possess. However, due to humans' limited mobility, we assumed a small tracking scene (such as the size of a neighbourhood or similar venues).

- *Part 2: Vehicular tracking.* We extend the tracking scene to consider vehicles, which have much rapid (but more constrained) mobility patterns than pedestrians. In this case, our tracking scene is restricted to a city size area. We consider both online (with mobility prediction) and offline forensic tracking.
- *Part 3: Multimodal tracking.* In this part, we further extend the tracking scene and consider a combination of parts 1 and 2. In multi-modal tracking we try to track the targets over a large metropolitan area assuming that target entities may opt to use different modes of transportation such as vehicular and other public transportation means (buses, trains etc.). We only considered offline tracking because it is more applicable in the real world, where online multi-modal tracking may be infeasible or unrealistic having such large tracking area.

In this final chapter, we discuss how this thesis can be extended and provide ideas for future work that we believe would be interesting to investigate.

## 8.1 Offline Pedestrian Forensic Tracking

As discussed above, throughout the thesis, we developed and proposed various forensic tracking algorithms for three different tracking environments: pedestrian (chapter 4), vehicular (chapters 5 and 6) and multi-modal (chapter 7). While we investigated both online and offline forensic tracking in the vehicular setting, we only considered online forensic tracking in the pedestrian setting and only offline forensic tracking in the multi-modal setting. Thus, a logical extension to this thesis is to consider the missing scenarios, namely offline pedestrian forensic tracking and online multi-modal forensic tracking. However, it is clear that the online forensic tracking in a multi-modal setting (for a sufficiently large, realistic) scenario becomes unmanageable very quickly, so we doubt its relevance and feasibility. Offline pedestrian forensic tracking, on the other hand, may be more interesting and certainly would have more applications than the online one. This kind of tracking is, nevertheless, more challenging than the offline vehicular forensic tracking due to the unpredictability of pedestrian movement and thus will need more careful modelling in order to reconstruct the traces accurately.

## 8.2 Advanced Bayesian-based Trace Reconstruction

In chapter 3 we discussed the Bayesian approach for trace reconstruction, then in chapter 6 we demonstrated how to use simple Bayesian inference for this purpose in a vehicular setting. However, in chapter 6, we only considered the worst case scenario,

where we do not have any external information about the scene in which the target's (incomplete) traces were obtained. In that case, simple Bayesian inference was sufficient and probably the only option. However, if we happen to have access to more information about the scene, we will have more variables and then the more powerful Bayesian Networks (with their dynamic and fuzzy extensions) can be used. It will, therefore, be interesting to create such scenarios and develop a Bayesian Network to investigate how accurately traces can be reconstructed, but we note that for a large number of variables, Bayesian networks become exponentially complex, as discussed in chapter 3 (with their dynamic and fuzzy extensions, complexity grows even faster).

### 8.3 Tracking based on Social Networking

This thesis has been exclusively concerned with the physical tracking of targets from online passive tracking to offline tracking and trace reconstruction. The contributed algorithms carried out the tracking based on physical observations, but sometimes such physical traces may not be available/accessible due to resource constraints. In this case, logical tracking may be pursued instead. The theory of *social networking* seems to provide an interesting tool for this purpose. In this theory, individuals are represented by nodes that are connected to each other by ties, such as friendship, business etc., forming a network of interconnected nodes. This network can then be analysed in many ways and various hypotheses can be made based on connections between the nodes. Recent work [26] used such approach to investigate child sex trafficking networks and the result seems promising. Similar approach can be used for different forensic purposes and would make an interesting extension to the work presented in this thesis.

### 8.4 Crime Reconstruction

The main contribution of this thesis is the investigation of how to reconstruct traces belonging to particular individuals, considering different environments and settings. While this will certainly assist in most criminal investigations (as we discussed extensively in chapter 1 and throughout the thesis), trace reconstruction is *not* synonym to the more general field of crime reconstruction. Also known as crime scene reconstruction, crime reconstruction is the process of reconstructing the sequence of the events leading to a crime by making hypotheses based on evidence collected from the crime scene or obtained by external sources. Traditionally, these evidence were mainly biological substances that were inadvertently left by the offenders at the crime scene and later found by police investigators. While such substances still provide significant

evidence in most criminal investigations, today digital evidence and traces are also common leftover pieces of evidence. Crime reconstruction is not confined to the reconstruction of events that took place before or during the crime, but also in most cases considers the events that occurred after or as a result of the crime, which also usually provide significant evidence. Therefore, trace reconstruction can indeed be considered a subset of crime reconstruction. Clearly, then, it would be interesting to investigate other aspects of crime reconstructions that can complement trace reconstruction.

## 8.5 Multi-modal Trace Reconstruction System

In chapter 7 we proposed a complete multi-modal trace reconstruction system. Although we provided a theoretical complexity evaluation of the reconstruction algorithm and showed that it is both optimal and complete, it was not possible to actually implement and practically test this system due to resource and time constraints. The reconstruction algorithm uses several mobility models, which we especially proposed and tailored for forensic analysis. These models resemble other existing mobility models except that they are simpler since in our scenario we are not concerned about the various microscopic details that conventional mobility models usually take great care to describe precisely. This, however, does not mean that these models do not need to be evaluated independently, but such evaluation will have to be undertaken alongside the evaluation of our multi-modal trace reconstruction system since they are part of it. In other words, we developed these models especially to be used with our trace reconstruction system, and as such they cannot replace the normal mobility models in applications where the conventional mobility models are usually used (e.g., for evaluating protocols) because they would not generate precise node mobility traces, rather they would only generate the time delay that simulated nodes would generally take in traversing a particular area or route. This behaviour, however, is useless in most mainstream (non-forensic) applications using mobility models. Thus, we felt that evaluating these mobility models (which, for the sake of distinction, we called “mobility delay models”) without actually using them in an implementation of our trace reconstruction system, has little value as they may not be used elsewhere. Therefore, an immediate extension to this thesis is to implement our multi-modal trace reconstruction system and evaluate it in practice, but this will likely take quite some time given the detailed structure of the system. We believe that this system has the potential to go beyond academic research to find its way to the real world and be an important tool which law enforcement can use in real modern criminal investigations.

## 8.6 Live Vehicular Forensics

In this thesis, we considered the reconstruction of location traces of target individuals. However, in [5], it was shown that significant evidence can be extracted from modern intelligent vehicles due to the rich set of sensors they are equipped with. Such evidence may not only include location traces, but also video and audio evidence. In [5], each sensor was considered individually to discuss the evidence that can be obtained from them. A possible extension to this thesis is to consider this new source of evidence and characterise/fuse such sensor data sources. Most of this data is, however, volatile, thus it would be interesting to investigate what kind of non-volatile data<sup>1</sup> that common vehicular systems preserve and store in-memory. Our expectation is that most data of this type is not relevant to the forensic behavioural analysis of individuals. However, some automotive functions may be capable of storing useful information as part of their normal operation, possibly with user interaction. For example, most navigation systems maintain history records of previous destinations entered by the user in addition to a *favourite locations* list and a *home* location bookmark configured by the user; such data and configurations are likely to be non-volatile and can be easily retrieved at later times. Moreover, these systems may also contain information on *intended* movement, which is of particular interest if it can be communicated to investigators in real-time, that will enable anticipating target movements.

## 8.7 Final Remarks

In this thesis, we investigated and studied a computational forensic discipline that we believe has significant relevance to modern crime investigation and prevention technologies, we call this discipline “forensic tracking”. Forensic tracking and surveillance is the process of investigating and reconstructing the location of a target (or targets) for forensic purposes. Like conventional tracking, forensic tracking can either be online or offline. We studied and elaborated on both types while considering different environments and settings. We proposed algorithms for passive online pedestrian and vehicular forensic tracking, then proceeded to consider vehicular offline forensic tracking and finally concluded with offline multi-modal forensic tracking, which is the most challenging as it combines both vehicular and pedestrian settings. This thesis demonstrates the practical importance and relevance of forensic tracking and surveillance and envisions that this discipline will become an integral part of modern criminal investigations, while still posing interesting academic research for years to come.

---

<sup>1</sup>Data captured by the Event Data Recorder (EDR) is non-volatile, but it is not always interesting or relevant for forensic investigations.

# Bibliography

- [1] S. Al-Kuwari and S. D. Wolthusen. A Survey of Forensic Localization and Tracking Mechanisms in Short-Range and Cellular Networks. In *International Conference on Digital Forensics and Cybercrime (ICDF2C '09)*, volume 31 of *LNICST*, pages 19–32. Springer-Verlag, 2009. [9](#), [23](#)
- [2] S. Al-Kuwari and S. D. Wolthusen. Passive Ad-Hoc Localization and Tracking in Short-Range Communication. In *International Conference on Next Generation Wireless Systems (NGWS '09)*, ISBN 3838353463. LAB Lambert Academic Publisher, 2009. [39](#)
- [3] S. Al-Kuwari and S. D. Wolthusen. Algorithms for Advanced Clandestine Tracking in Short-Range Ad Hoc Networks. In *International ICST Conference on Security and Privacy in Mobile Information and Communication Systems (MobiSec '10)*, volume 46 of *LNICST*, pages 67–79. Springer-Verlag, 2010. [39](#)
- [4] S. Al-Kuwari and S. D. Wolthusen. Forensic Tracking and Mobility Prediction in Vehicular Networks. In *IFIP WG 11.9 International Conference on Digital Forensics*, volume 337 of *Advances in Digital Forensics VI*, pages 91–105. Springer-Verlag, 2010. [70](#)
- [5] S. Al-Kuwari and S. D. Wolthusen. On the Feasibility of Carrying out Live Real-Time Forensics for Modern Intelligent Vehicles. In *International ICST Conference on Forensic Applications and Techniques in Telecommunications, Information and Multimedia (e-Forensics '10)*, volume 56 of *LNICST*, pages 218–234. Springer-Verlag, 2010. [142](#)
- [6] S. Al-Kuwari and S. D. Wolthusen. Probabilistic Vehicular Trace Reconstruction Based on RF-Visual Data Fusion. In *Joint IFIP TC6 and TC11 Conference on Communications and Multimedia Security (CMS '10)*, volume 6109 of *LNCS*, pages 16–27. Springer-Verlag, 2010. [89](#)

- [7] S. Al-Kuwari and S. D. Wolthusen. Algorithmic Approach for Clandestine Localisation and Tracking in Short-Range Environments. *International Journal of Communication Networks and Distributed Systems (IJCND)*, 2011. (To appear). [39](#)
- [8] S. Al-Kuwari and S. D. Wolthusen. Fuzzy Trace Validation: Toward an Offline Forensic Tracking Framework. In *6th IEEE Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE '11)*. IEEE Press, 2011. (To appear). [23](#), [107](#)
- [9] S. Al-Kuwari and S. D. Wolthusen. Multi-Modal Trace Reconstruction: an Offline Forensic Tracking Approach. In *IFIP WG 11.9 International Conference on Digital Forensics*, 2012. (To appear). [107](#)
- [10] S. Aparicio, J. Perez, A. Bernardos, and J. Casar. A Fusion Method Based on Bluetooth and WLAN Technologies for Indoor Location. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI '08)*, pages 487–491, 2008. [19](#)
- [11] S. Aparicio, P. Tarrio, J. Perez, A. Bernardos, and J. Casar. An Indoor Location Method Based on a Fusion Map Using Bluetooth and WLAN Technologies. In *International Symposium on Distributed Computing and Artificial Intelligence (DCAI '08)*, volume 50 of *ASC*, pages 702–710. Springer-Verlag, 2009. [19](#)
- [12] J. Ash and L. Potter. Robust System Multiangulation Using Subspace Methods. In *International Conference on Information Processing in Sensor Networks (IPSN '07)*, pages 61–68, 2007. [17](#)
- [13] F. Bai, N. Sadagopan, and A. Helmy. The IMPORTANT Framework for Analyzing the Impact of Mobility on Performance of Routing Protocols for Ad Hoc Networks. *Elsevier Ad Hoc Networks*, 1:383–403, 2003. [34](#)
- [14] S. Barakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On Map-Matching Vehicle Tracking Data. In *International Conference on Very Large Databases (VLDB '05)*, pages 853–864. VLDB Endowment, 2005. [71](#), [91](#)
- [15] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded Up Robust Features. In *Computer Vision (ECCV '06)*, 2006. [92](#)
- [16] A. Benslimane. Localization in Vehicular Ad Hoc Networks. In *Systems Communications (ICW '05)*, pages 19–25, 2005. [71](#)

- [17] N. Borenovic, I. Simic, M. Neskovic, and M. Petrovic. Enhanced Cell-ID + TA GSM Positioning Technique. In *EUROCON '05*, volume 2, pages 1176–1179, 2005. [17](#)
- [18] A. Boukerche, H. Oliveira, and E. Nakamura. Vehicular Ad Hoc Networks: A New Challenge for Localization-based Systems. *Computer communications*, 31(12):2838–2849, 2008. [71](#)
- [19] A. Calbi, L. Marcenaro, and C. Regazzoni. Dynamic Scene Reconstruction for 3D Virtual Guidance. In *Knowledge-based Intelligent Information and Engineering Systems (KES '06)*, pages 179–186, 2006. [90](#)
- [20] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002. [31](#)
- [21] N. Cartner, C. Messer, and A. Rathi. Flow Theory: A State of the Art Report - Revised Monograph on Traffic Flow Theory. Technical report, Turner-Fairbank Highway Research Center, 2001. [34](#)
- [22] A. Catovic and Z. Sahinoglu. Hybrid TOA/RSS and TDOA/RSS Location Estimation Schemes for Short-Range Wireless Networks. *Bechtel Telecommunication Technical Journal (BTTJ)*, 2(2):77–84, 2004. [20](#)
- [23] A. Catovic and Z. Sahinoglu. The Cramer-Rao Bounds of Hybrid TOA/RSS and TDOA/RSS Location Estimation Schemes. *IEEE Communications Letters*, 8(10):626–628, 2004. [20](#)
- [24] Y. Chan, W. Tsui, and H. So. Time-of-Arrival Based Localization Under NLOS Conditions. *IEEE Transactions on Vehicular Technology*, 55(1):17–24, 2006. [12](#)
- [25] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005. [122](#)
- [26] E. Cockbain, H. Brayley, and G. Laycock. Exploring Internal Child Sex Trafficking Networks Using Social Network Analysis. *Policing*, 5(2):144–157, 2011. [140](#)
- [27] C. O. Conaire, K. Fogarty, C. Brennan, and N. E. O'Connor. User Localisation using Visual Sensing and RF Signal Strength. In *Workshop on Applications, Systems, and Algorithms for Image Sensing (ImageSese '08)*, 2008. [91](#)

- [28] L. Cong and W. Zhuang. Hybrid TDOA/AOA Mobile User Location for Wideband CDMA Cellular Systems. *IEEE Transactions on Wireless Communications*, 1(3):439–447, 2002. [21](#)
- [29] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2009. [135](#)
- [30] David Johnson and David Maltz. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, pages 153–181. Kluwer Academic Publishers, 1996. [31](#)
- [31] V. Davies. Evaluating Mobility Models Within Ad Hoc Networks. Master’s thesis, Colorado School of Mines, 2000. [34](#)
- [32] A. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977. [128](#)
- [33] E. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271, 1959. [125](#)
- [34] G. Djuknic and R. Richton. Geolocation and Assisted GPS. *Computer*, 34(2):123–125, 2001. [18](#)
- [35] A. Domazetovic, J. Greenstein, B. Mandayam, and I. Seskar. Propagation Models for Short-Range Wireless Channels with Predictable Path Geometries. *IEEE Transactions on Communications*, 53(7):1123–1126, 2005. [22](#)
- [36] D. Engelhart, C. Barrett, and M. Morin. A Spatial Analysis of Mobility Models: Application to Wireless Ad Hoc Network Simulation. In *ANSS ’04*, pages 35–42, 2004. [79](#)
- [37] Federal Communications Commission (FCC). *OET Bulletin no. 71: Guidelines for Testing and Verifying the Accuracy of E911 Location Systems*, 2000. [17](#)
- [38] M. Fewell. Area of Common Overlap of Three Circles. Technical Report DSTON-TN-0722, Maritime Operations Division, Defence Science and Technology Organisation, 2006. [15](#)
- [39] M. Fiore, J. Harri, F. Fethi, and C. Bonnet. Vehicular Mobility Simulation for VANETs. In *Annual Simulation Symposium (ANSS ’07)*, 2007. [102](#)

- [40] M. Fiore, J. Harri, F. Filali, and C. Bonnet. Understanding Vehicular Mobility in Network Simulation. In *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS '10)*, pages 1–6, 2007. [34](#)
- [41] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian Filtering for Location Estimation. *IEEE Pervasive Computing*, 2(3):24–33, 2003. [28](#), [46](#)
- [42] K. Franke and S. N. Srihari. Computational Forensics: Towards Hybrid-Intelligent Crime Investigation. In *International Symposium on Information Assurance and Security (IAS '07)*, pages 383–386, 2007. [3](#)
- [43] H. Friis. A Note on a Simple Transmission Formula. *Proceedings of the IRE*, 34(5):254–256, 1946. [46](#), [73](#)
- [44] S. Frühwirth-Schnatter. On Fuzzy Bayesian Inference. *Fuzzy Sets Systems*, 60:41–58, 1993. [30](#)
- [45] D. Gazis, R. Herman, and R. Rothery. Nonlinear Follow-the-Leader Models of Traffic Flow. *Operational Research*, 9(4):545–567, 1961. [34](#)
- [46] Z. Ghahramani. Learning Dynamic Bayesian Networks. In *Adaptive Processing of Sequences and Data Structures, International Summer School on Neural Networks*, pages 168–197, 1998. [30](#)
- [47] W. Gilks, S. Richardson, and D. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC Interdisciplinary Statistics, 1995. [129](#)
- [48] S. Greenhill and S. Venkatesh. Virtual Observers in a Mobile Surveillance System. In *Annual ACM International Conference on Multimedia (MULTIMEDIA '06)*, pages 579–588, 2006. [91](#)
- [49] J. Gross and J. Yellen. *Graph Theory and Its Applications*. Discrete Mathematics and Its Application. Chapman and Hall/CRC, 2nd edition, 2005. [135](#)
- [50] F. Gustafsson and F. Gunnarsson. Positioning Using Time-Difference of Arrival Measurements. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, volume 6, pages 553–556, 2003. [12](#)
- [51] D. Hall and J. Llinas. An Introduction to Multisensor Data Fusion. In *Proceedings of the IEEE*, volume 85, pages 6–23, 1997. [19](#)
- [52] F. Harary. *Graph Theory*. Westview Press, 1994. [135](#)

- [53] J. Harri, F. Filali, and C. Bonnet. Mobility Models for Vehicular Ad Hoc Networks: a Survey and Taxonomy. *IEEE Communications Surveys and Tutorials*, 11(4):19–41, 2009. [120](#)
- [54] J. Harri, M. Fiore, F. Fethi, and C. Bonnet. VanetMobiSim: Generating Realistic Mobility Patterns for VANETs. In *ACM International Workshop on Vehicular Ad Hoc Networks (VANET '06)*, 2006. [82](#), [102](#)
- [55] N. Hartsfield and G. Ringel. *Pearls in Graph Theory: A Comprehensive Introduction*. Dover Publications, 2005. [135](#)
- [56] D. Helbing. A Fluid Dynamic Model for the Movement of Pedestrians. *Complex Systems*, 6:391–415, 1992. [31](#)
- [57] D. Helbing, I. Farkas, and T. Vicsek. Simulating Dynamical Features of Escape Panic. *Nature*, 407(6803):487–490, 2000. [32](#)
- [58] D. Helbing, I. Frakas, P. Molnar, and T. Vicsek. *Pedestrian and Evacuation Dynamics*, chapter Simulation of Pedestrian Crowds in Normal and Evacuation Situations, pages 21–58. Springer, Berlin, 2002. [32](#)
- [59] D. Helbing and P. Molnar. Social Force Model for Pedestrian Dynamics. *Physical Review E*, 51(5):4282–4286, 1995. [32](#), [118](#)
- [60] J. Hightower and G. Borriello. A Survey and Taxonomy of Location Systems for Ubiquitous Computing. *IEEE Computer*, 34:57–66, 2001. [10](#)
- [61] J. Hightower and G. Borriello. Location Sensing Techniques. Technical Report 01-07-01, University of Washington, Department of Computer Science and Engineering, 2001. [10](#)
- [62] J. Hightower and G. Borriello. Location Systems for Ubiquitous Computing. *Computer*, 34(8):57–66, 2001. [10](#)
- [63] O. Hjelle and M. Daehlen. *Triangulations and Applications*. Springer, 2006. [13](#)
- [64] A. K. M. Hossain and W.-S. Soh. A Comprehensive Study of Bluetooth Signal Parameters for Localization. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '07)*, 2007. [21](#)
- [65] C. Hsin-Yuan and C. Tung-Yi. Hybrid TDOA/AOA Mobile User Location with Artificial Neural Networks. In *IEEE International Conference on Networking, Sensing and Control (ICNSC '08)*, pages 847–852, 2008. [21](#)

- [66] L. Hu and D. Evans. Localization for Mobile Sensor Networks. In *Annual international conference on Mobile computing and networking (MobiCom '04)*, pages 45–57, 2004. [11](#)
- [67] H. Huang, P. Lue, M. Li, D. Li, X. Li, W. Shu, and M. Wu. Performance Evaluation of SUVnet With Real-Time Traffic Data. *IEEE Transactions on Vehicular Technology*, 56(6):3381–3396, 2007. [34](#)
- [68] I. Hwang, H. Balakrishnan, K. Roy, and C. Tomlin. Multiple-target Tracking and Identity Management in Clutter, with Application to Aircraft Tracking. In *American Control Conference (ACC '04)*, 2004. [26](#)
- [69] Information Sciences Institute, University of Southern California. *Network Simulator 2 (NS-2)*. ([www.isi.edu/nsnam/ns](http://www.isi.edu/nsnam/ns)). [82](#), [102](#)
- [70] B. Jiang, B. Ravindran, and H. Cho. Energy Efficient Sleep Scheduling in Sensor Networks for Multiple Target Tracking. In *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '08)*, volume 5067 of *LNCs*, pages 498–509. Springer-Verlag, 2008. [25](#)
- [71] F. Kargle and A. Bernauer. The COMPASS Location System. In *First International Workshop on Location and Context Awareness (LoCA 2005)*, volume 3479 of *LNCs*. Springer-Verlag, 2005. [19](#)
- [72] F. Kargle, G. Dannhäuser, S. Schlott, and J. Nagler-Ihle. Semantic Information Retrieval in the COMPASS Location System. In *Ubiquitous Computing Systems (UCS 2006)*, volume 4239 of *LNCs*. Springer-Verlag, 2006. [20](#)
- [73] H. Karimi, T. Conahan, and D. Roongpiboonsopit. A Methodology for Predicting Performances of Map-Matching Algorithms. In *International Symposium on Web and Wireless Geographical Information Systems (W2GIS '06)*, pages 202–213, 2006. [91](#)
- [74] H. Kim, E. Kim, and K. Han. An Energy Efficient Tracking Method in Wireless Sensor Networks. In *Next Generation Teletraffic And Wired Wireless Advanced Networking (NEW2AN '06)*, volume 4003 of *LNCs*, pages 278–286. Springer-Verlag, 2006. [25](#)
- [75] W. Kim, K. Mechtov, J.-Y. Choi, and S. Ham. On Target Tracking with Binary Proximity Sensors. In *International Symposium on Information Processing in Sensor Networks (IPSN '05)*, 2005. [25](#)

- [76] T. Kleine-Ostmann and A. Bell. A Data Fusion Architecture for Enhanced Position Estimation in Wireless Networks. *IEEE Communications Letters*, 5(8):343–345, 2001. [19](#)
- [77] T. Kos, M. Grgic, and J. Kitarovic. Location Technologies for Mobile Networks. In *EURASIP '07*, pages 319–322, 2007. [17](#)
- [78] D. Krajzewicz, G. Hertkorn, C. Rssel, and P. Wagner. SUMO (Simulation of Urban MObility): An Open-Source Traffic Simulation. In *The Middle Eastern Modelling and Simulation MultiConferences (MESM '02)*, 2002. (sumo.sourceforge.net). [122](#)
- [79] S. KrauB. *Microscopic Modeling of Traffic Flow: Investigation of Collision Free vehicle Dynamics*. PhD thesis, Mathematisches Institut, Universitat zu Koln, 1998. [34](#)
- [80] T. I. Lakoba, D. J. Kaup, and N. M. Finkelstein. Modifications of the Helbing-Molnar-Varkas-Vicsek Social Force Model for Pedestrian Evolution. *Simulation*, 81(5):339–352, 2005. [31](#)
- [81] A. H. Land and A. G. Doig. An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28(3):497–520, 1960. [130](#)
- [82] M. Lee, K. Kim, and H. Lee. *Information Hiding*, chapter Forensic Tracking Watermarking against In-theater Piracy, pages 117–131. Springer-Verlag, 2009. [23](#)
- [83] H. Leung, H. Zhijian, and M. Balachette. Evaluation of Multiple Radar Target Trackers in Stressful Environments. *IEEE Transactions on Aerospace and Electronic Systems*, 35(2):663–674, 1999. [26](#)
- [84] M. Lighthill and G. Whitham. On Kinematic Waves. II. A Theory of Traffic Flow on Long Crowded Roads. *Proc. of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 229(1178):317–345, 1955. [34](#)
- [85] J. Llinas and D. Hall. An Introduction to Multi-sensor Data Fusion. In *IEEE International Symposium on Circuits and Systems (ISCAS '98)*, 1998. [19](#)
- [86] C. Lochert, B. Scheuermann, and M. Mauve. *VANET: Vehicular Applications and Inter-Networking Technologies*, chapter Information Dissemination in VANETs, pages 49–80. John Wiley and Sons, Ltd., 2010. [74](#)

- [87] M. Luber, J. Stork, G. Tipaldi, and K. Arras. People Tracking with Human Motion Predictions from Social Forces. In *IEEE International Conference on Robotics and Automation (ICRA '10)*, 2010. 32
- [88] R. Lue, O. Chen, and L. Tu. Node Localization through Data Fusion in Sensor Network. In *International Conference on Advanced Information Networking and Applications (AINA '05)*, pages 337–342, 2005. 20
- [89] J. Luo and J. Hubaux. A Survey of Inter-Vehicle Communication. Technical Report IC/2004/24, School of Computer and Communication Science, EPFL, 2004. 72
- [90] L. Mailaender. On the Geolocation Bounds for Round-Trip Time-of-Arrival and All Non-Line-of-Sight Channels. *EURASIP Journal on Advances in Signal Processing*, 2008, 2008. 12
- [91] J. Mansell and W. Riley. Vehicle Tracking and Security System, 1993. 71
- [92] G. Mao, BarışFidan, and B. Anderson. Wireless Sensor Network Localization Techniques. *The International Journal of Computer and Telecommunications Networking*, 51(10):2529–2553, 2007. 11
- [93] L. Mihaylova, D. Angelova, S. Honary, D. Bull, C. Canagarajah, and B. Ristic. Mobility Tracking in Ceullar Network Using Particle Filtering. *IEEE Transactions on Wireless Communciations*, 6(10):3589–3599, 2007. 25
- [94] C. Miyajima, Y. Nishiwaki, K. Ozawa, T. Wakita, K. Itou, K. Takeda, and F. Itakura. Driver Modeling Based on Driving Behavior and Its Evaluation in Driver Identification. *Proceedings of the IEEE*, 95(2):427–437, 2007. 101
- [95] G. Mizusawa. Performance of Hyperbolic Position Location Techniques for Code Division Multiple Access. Master’s thesis, Virginia Polytechnic Institute and State University, 1996. 13
- [96] F. Mondinelli and Z. Kovacs-Vajna. Self-Localizing Sensor Network Architectures. *IEEE Transactions on Instrumentation and Measurement*, 53(2):277–283, 2004. 11, 45
- [97] J. Morgan-Owen and T. Johnston. Differential GPS Positioning. *Electronics & Communication Engineering Journal*, 7(1):11–21, 1995. 19
- [98] N. Nasser. Automatic Location Systems for Mobile Phones. *Arab Research Institute in Sciences & Engineering (ARISER)*, 2(2):53–59, 2008. 18

- [99] N. Nilsson. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, 1971. [135](#)
- [100] M. Olama, S. Djouadi, and C. Charalambous. Position and Velocity Tracking in Cellular Networks Using Particle and Kalman Filtering with Comparison. In *IEEE Conference on Decision and Control (CDC '06)*, pages 1315–1320, 2006. [25](#)
- [101] E. O'Neill, V. Kostakos, T. Kindberg, A. Fatah gen. Schiek, A. Penn, D. Stanton Fraser, and T. Jones. Instrumenting the City: Developing Methods for Observing and Understanding the Digital Cityscape. In *International Conference on Ubiquitous Computing (UbiComp '06)*, pages 315–332, 2006. [40](#)
- [102] S. Pandey and P. Agrawal. A Survey on Localization Techniques for Wireless Networks. *Journal of the Chinese Institute of Engineers*, 29(7):1125–1148, 2006. [10](#)
- [103] N. Patwari, J. Ash, S. Kyperountas, A. Hero, R. Moses, and N. Correal. Locating the Nodes: Cooperative Localization in Wireless Sensor Networks. *IEEE Signal Processing Magazine*, 22(4):54–69, 2005. [13](#), [22](#)
- [104] S. Pemmaraju and S. Skiena. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Cambridge University Press, 2003. [135](#)
- [105] Z. Ping, L. Ling-yan, and S. Hao-shan. A Hybrid Location Algorithm Based on BP Neural Networks for Mobile Position Estimation. *IJCSNS International Journal of Computer Science and Network Security*, 6(7A):162–167, 2006. [21](#)
- [106] PTV system GmbH. *VISSIM*. ([www.ptvamerica.com](http://www.ptvamerica.com)). [122](#)
- [107] A. Raza, S. Hameed, and T. Macintyre. *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering*, chapter Global Positioning System - Working and its Applications, pages 448–453. Springer Netherlands, 2008. [18](#)
- [108] D. Reid. An Algorithm for Tracking Multiple Targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979. [26](#)
- [109] P. Rong and L. Sichitiu. Angle of Arrival Localization for Wireless Sensor Networks. In *Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON '06)*, volume 1, pages 374–382, September 2006. [13](#)

- [110] T. Roos, P. Myllymäki, and H. Tirri. A Statistical Modeling Approach to Location Estimation. *IEEE Transactions on Mobile computing*, 1(1):59–69, 2002. [11](#)
- [111] S. Roychowdhury and W. Pedrycz. A Survey of Defuzzification Strategies. *International Journal of Intelligent Systems*, 16(6):679–695, 2001. [127](#)
- [112] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009. [135](#)
- [113] A. Saha and D. Johnson. Modeling Mobility for Vehicular Ad Hoc Networks. In *ACM International Workshop on Vehicular Ad Hoc Networks (VANET '04)*, 2004. [33](#)
- [114] M. Saxena, P. Gupta, and B. Jain. Experimental Analysis of RSSI-based Location Estimation in Wireless Sensor Networks. In *International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, pages 503–510, 2008. [12](#)
- [115] A. Schadschneider. Cellular Automaton Approach to Pedestrian Dynamics - Theory. *Pedestrian and Evacuation Dynamics*, pages 75–86, 2002. [31](#)
- [116] J. Seybold. *Introduction to RF Propagation*. Wiley-Interscience, 2005. [21](#), [22](#), [46](#)
- [117] Y. Shang, H. Shi, and A. Ahmed. Performance Study of Localization Methods for Ad Hoc Sensor Networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS '04)*, pages 184–193, 2004. [17](#)
- [118] X. Sheng, Y.-H. Hu, and P. Ramanathan. Distributed Particle Filter with GMM Approximation for Multiple Targets Localization and Tracking in Wireless Sensor Network. In *International Symposium on Information Processing in Sensor Networks (IPSN '05)*, 2005. [25](#)
- [119] W. Simpson. *PPP Challenge Handshake Authentication Protocol (CHAP)*. RFC 1994, August 1996. [58](#)
- [120] G. Sinan. A Survey on Wireless Position Estimation. *Wirel. Pers. Commun.*, 44(3):263–282, 2008. [10](#)
- [121] J. Singh, U. Madhow, R. Kumar, S. Suri, and R. Cagley. Tracking Multiple Targets Using Binary Proximity Sensors. In *International Conference on Information OProcessing in Sensor Networks (IPSN '07)*, 2007. [25](#)

- [122] M. Tan, G.-L. Tian, and K. W. Ng. *Bayesian Missing Data Problems: EM, Data Augmentation and Noniterative Computation*. CRC Press, 2009. [90](#)
- [123] A. Tanenbaum. *Computer Networks*. Pearson Education Ltd., 2003. [96](#)
- [124] M. Tanner and W. Wong. The Calculation of Posterior Distributions by Data Augmentation. *Journal of the Americal Statistical Association*, 82(398):528–540, 1987. [128](#)
- [125] M. Treiber, A. Hennecke, and D. Helbing. Congested Traffic States in Empirical Observations and Microscopic Simulations. *Physical Review E*, 62(2):1805–18024, 2000. [72](#), [102](#)
- [126] U.S. Census Beureau. *Topologically Integrated Geographic Encoding and Referencing (TIGER)*. ([www.census.gov/geo/www/tiger](http://www.census.gov/geo/www/tiger)). [85](#)
- [127] S. Venkatraman and J. Caffery. Hybrid TOA/AOA Techniques for Mobile Location in Non-Line-Of-Sight Environments. In *IEEE Wireless Communications and Networking Conference (WCNC '04)*, volume 1, pages 274–278, 2004. [20](#)
- [128] N. Vyahhi and S. Bakiras. Tracking Moving Objects in Anonymized Trajectories. In *International Conference on Database and Expert Systems Applications (DEXA '08)*, volume 5181 of *LNCS*, pages 158–171. Springer-Verlag, 2008. [26](#)
- [129] A. Wahab, C. Quek, C. K. Tan, and K. Takeda. Driving Profile Modeling and Recognition Based on Soft Computing Approach. *IEEE Transactions on Neural Networks*, 20(4):563–582, 2009. [104](#)
- [130] M. Winter and G. Taylor. A Modular Neural Network Approach to Improve Map-Matched GPS Positioning. In *Web and Wireless Geographical Information Systems (W2GIS '06)*, pages 76–89, 2006. [91](#)
- [131] Y. Xu and W. Lee. On Localized Prediction for Power Efficient Object Tracking in Sensor Networks. In *International Conference on Distributed Computing Systems Workshops (ICDCSW '03)*, page 434, 2003. [24](#)
- [132] H. Yang and B. Sikdar. A Protocol for Tracking Mobile Tragetts Using Sensor Networks. In *IEEE International Workshop on Sensor Network Protocols and Applications (SNPA '03)*, pages 71–81, 2003. [24](#)
- [133] J. Yoon, M. Liu, and B. Noble. Random Waypoint Considered Harmful. In *INFOCOM '03*, volume 2, pages 1312–1321, 2003. [31](#)

- [134] L. Zadeh. Fuzzy Sets. *Information and Control*, 8(3):338–353, 1965. [125](#)
- [135] Z. Zaidi and B. Mark. Real-Time Mobility Tracking Algorithms for Cellular Networks Based on Kalman Filtering. *IEEE Transactions on Mobile Computing*, 4(2):195–208, 2005. [25](#)
- [136] C. Zhang, J. Liu, S. Liu, and W. Li. Research on Improving TDOA Location Accuracy Based on Data Fusion. In *IEEE 6th Circuits and Systems Symposium on Emerging Technologies*, volume 2, pages 761–764, 2004. [20](#)