



Michigan Technological University  
*Create the Future* Digital Commons @ Michigan Tech

---

Dissertations, Master's Theses and Master's  
Reports - Open

Dissertations, Master's Theses and Master's  
Reports

---

2014

## USING PROBABILISTIC GRAPHICAL MODELS TO DRAW INFERENCES IN SENSOR NETWORKS WITH TRACKING APPLICATIONS

Lufeng Shi  
*Michigan Technological University*

Follow this and additional works at: <https://digitalcommons.mtu.edu/etds>



Part of the [Computer Engineering Commons](#)

Copyright 2014 Lufeng Shi

---

### Recommended Citation

Shi, Lufeng, "USING PROBABILISTIC GRAPHICAL MODELS TO DRAW INFERENCES IN SENSOR NETWORKS WITH TRACKING APPLICATIONS", Dissertation, Michigan Technological University, 2014.  
<https://doi.org/10.37099/mtu.dc.etds/752>

Follow this and additional works at: <https://digitalcommons.mtu.edu/etds>



Part of the [Computer Engineering Commons](#)

USING PROBABILISTIC GRAPHICAL MODELS TO DRAW INFERENCES IN  
SENSOR NETWORKS WITH TRACKING APPLICATIONS

By

Lufeng Shi

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Computer Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2014

© 2014 Lufeng Shi

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Computer Engineering.

Department of Electrical and Computer Engineering

Dissertation Advisor: *Dr. Jindong Tan*

Committee Member: *Dr. Jeffrey Burl*

Committee Member: *Dr. Warren Perger*

Committee Member: *Dr. Nilufer Onder*

Department Chair: *Dr. Daniel R. Fuhrmann*

# Contents

<b>List of Figures</b> . . . . .	<b>vii</b>
<b>List of Tables</b> . . . . .	<b>x</b>
<b>Preface</b> . . . . .	<b>xi</b>
<b>Acknowledgments</b> . . . . .	<b>xii</b>
<b>Abstract</b> . . . . .	<b>xiii</b>
<b>1 Introduction</b> . . . . .	<b>1</b>
1.1 Common Inference Method Used in Wireless Sensor Networks . . . . .	2
1.2 Main Challenges of Tracking and Detection in Wireless Sensor Network . . . . .	3
1.3 Thesis Organization . . . . .	4
<b>2 Background</b> . . . . .	<b>7</b>
2.1 Applications and Benefits of Wireless Sensor Networks . . . . .	7
2.2 Fundamental Inference Engine in Wireless Sensor Networks . . . . .	8
2.2.1 Bayesian Framework Formulation . . . . .	8
2.2.2 Bayesian Framework Based Tracking Algorithm . . . . .	9
2.3 Target Tracking in Sensor Networks Using Statistical Graphical Model . . . . .	10
2.3.1 Graph-based Tracking Algorithms . . . . .	11
2.3.2 Graph-based Event Boundary Detection Algorithms . . . . .	11
2.3.3 Cliques, Junction Trees and Markov Network . . . . .	11
2.4 Physical System of Our Ultrasonic Sensor Network . . . . .	13
2.4.1 Physical Properties of The Ultrasonic Sensor Nodes . . . . .	14
2.4.2 Self Localization of The Ultrasonic Sensor Network . . . . .	14
2.4.3 Timer Difference of Arrival Approach . . . . .	14
2.5 Synchronization Techniques in Sensor Networks . . . . .	15
2.5.1 Reference Broadcasting Synchronization (RBS) . . . . .	15
2.5.2 Romer's Time Synchronization Methd . . . . .	16
2.5.3 Post Facto Synchronization . . . . .	16
<b>3 Two-tier Tracking System</b> <sup>1</sup> . . . . .	<b>17</b>

---

<sup>1</sup>©2007 IEEE. Portions reprinted with permission, from **Lufeng Shi**, Zhijun Zhao, and Jindong Tan, "Near

3.1	Introduction . . . . .	17
3.2	Related Works . . . . .	18
3.3	Problem Definition . . . . .	20
3.4	Two-tier Tracking Framework in Chained-form Networks . . . . .	23
3.4.1	Tier One - Viterbi Detection . . . . .	23
3.4.2	Tier Two - Maximum Likelihood Estimation . . . . .	24
3.5	Two-tier Tracking in 2-Dimensional Network Topology . . . . .	27
3.5.1	Hexagon-shape Network Topology . . . . .	27
3.5.2	Grid-shape Network Topology . . . . .	27
3.6	Simulation . . . . .	29
3.6.1	Chained-form Network . . . . .	29
3.6.2	Hexagon-shape Network . . . . .	31
3.6.3	Grid-shape Network . . . . .	31
3.6.4	Tracking Errors . . . . .	32
3.6.5	Comparison to Similar Tracking Schemes . . . . .	33
3.7	Conclusion . . . . .	35
<b>4</b>	<b>Target Tracking in Markov Random Field <sup>2</sup></b> . . . . .	<b>36</b>
4.1	Introduction . . . . .	36
4.2	Related Works . . . . .	37
4.3	Problem Formulation . . . . .	40
4.4	Algorithm Formulation . . . . .	41
4.4.1	Chained-form network . . . . .	43
4.4.1.1	Node and link potential . . . . .	44
4.4.1.2	Message passing . . . . .	44
4.4.1.3	Inferences on clique tree . . . . .	45
4.4.2	Random distributed network . . . . .	45
4.5	Simulation . . . . .	47
4.5.1	Uniform Corridor . . . . .	48
4.5.2	Non-uniform Corridor . . . . .	48
4.5.3	Random sensor field . . . . .	48
4.6	Conclusion . . . . .	52
<b>5</b>	<b>Target Tracking with Grid-like Network Topology <sup>3</sup></b> . . . . .	<b>53</b>

---

*optimal two-tier target tracking in sensor networks*” , in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1993 - 1996, 2007. See Appendix for a copy of the copyright permission from IEEE.

<sup>2</sup>©2009 IEEE. Portions reprinted with permission, from **Lufeng Shi**, Jindong Tan, and Zhijun Zhao, “*Target tracking in sensor networks using statistical graphical models*” , in Proceedings of IEEE International Conference on Robotics and Biomimetics, pp. 2050 - 2055, 2009. See Appendix for a copy of the copyright permission from IEEE.

<sup>3</sup>©2009 IEEE. Portions reprinted with permission, from **Lufeng Shi**, and Jindong Tan, “*Distributive target tracking in sensor networks with a markov random field model*” , in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 854 - 859, 2009. See Appendix for a copy of the

5.1	Introduction . . . . .	53
5.2	Related Works . . . . .	54
5.3	Problem Formulation . . . . .	56
5.3.1	Graph Representation of Sensor Field . . . . .	56
5.3.2	Definition of Targets . . . . .	57
5.4	Tracking in A Graph With Ising Model . . . . .	58
5.5	Tracking in Randomly Distributed Sensor Network . . . . .	62
5.5.1	Grid-like Topology Construction . . . . .	62
5.5.2	Void Area in Grid-like Topology . . . . .	62
5.5.3	The Tracking Algorithm in A Grid-like Structure . . . . .	64
5.6	Simulations . . . . .	67
5.6.1	Cost Contrast . . . . .	68
5.6.1.1	Communication Cost . . . . .	68
5.6.1.2	Self healing . . . . .	72
5.7	Conclusion . . . . .	73
<b>6</b>	<b>Self Localization of Ultrasonic Sensor Network . . . . .</b>	<b>74</b>
6.1	Introduction . . . . .	74
6.2	Related Works . . . . .	75
6.3	Self-localization . . . . .	76
6.4	Ultrasonic Sensor Scheduling . . . . .	80
6.4.1	Sensor Scheduling Without Target Present . . . . .	80
6.4.2	Sensor Scheduling With Target Present . . . . .	81
6.4.3	Target Localization Within Triangles . . . . .	82
6.5	Simulation Results . . . . .	84
<b>7</b>	<b>Falling Detection in Elderly Homes with Radar Sensor Networks . . . . .</b>	<b>87</b>
7.1	Introduction . . . . .	87
7.2	System Architecture . . . . .	89
7.2.1	Automated falling detection system . . . . .	89
7.2.2	Hardware Platform . . . . .	90
7.2.2.1	Radar Sensor Board . . . . .	90
7.2.2.2	Sensor Deployment . . . . .	92
7.3	Local Detection . . . . .	93
7.3.1	Bumblebee Sensor Model . . . . .	93
7.4	Elderly Activity States . . . . .	94
7.4.1	Hidden Markov Model . . . . .	95
7.4.2	Parameter Assumptions . . . . .	96
7.5	Fall Detection Using HMM and Viterbi Algorithm . . . . .	98
7.6	Experimenting with Local Detection Decision . . . . .	99
7.7	Fusion Center . . . . .	100

---

copyright permission from IEEE.

<b>Conclusion</b> . . . . .	<b>105</b>
<b>References</b> . . . . .	<b>109</b>
<b>Appendix</b> . . . . .	<b>119</b>
Appendix A Copyright Permission for Chapter 3, 4 and 5 . . . . .	119

# List of Figures

2.1	Illustration of a clique . . . . .	12
2.2	Illustration of a junction tree . . . . .	13
3.1	Chained-form network topology . . . . .	21
3.2	One dimensional network topology with 2nd order connectivity derived from chained-form network . . . . .	21
3.3	Hexagon-shape network topology . . . . .	22
3.4	Grid-shape network topology . . . . .	22
3.5	Target detection hand off between sensors in tier 1 . . . . .	24
3.6	Cluster formed around the detection result obtained from tier 1 . . . . .	25
3.7	Message passing in hexagon-shape network with sensing region defined . . . . .	28
3.8	Message passing in grid-shape network with sensing region defined . . . . .	29
3.9	Detection inaccuracy due to grid-shape network . . . . .	30
3.10	Tracking in a chained-form network, $\sigma^2 = 0.01$ . . . . .	30
3.11	Tracking in a hexagon-shape network, $\sigma^2 = 0.01$ . . . . .	31
3.12	Tracking in a grid-shape network, $\sigma^2 = 0.01$ . . . . .	32
3.13	Tracking error for Chained-form network, hexagon-shape network and grid-shape network . . . . .	33
3.14	Triplet circle intersection (TCI) method . . . . .	34
4.1	Challenges in random distributed sensor networks . . . . .	41
4.2	Example of network topology and associated clique trees . . . . .	42
	(a) Possible clique tree structure derived from topology . . . . .	42
	(b) Alternative clique tree structure . . . . .	42
4.3	Clique trees for uniform and non-uniform chained-form regions . . . . .	43
	(a) Uniform distributed corridor . . . . .	43
	(b) Non-uniform distributed corridor . . . . .	43
4.4	Corridor obtained in a random distributed network . . . . .	47
	(a) Topology of a random distributed network . . . . .	47
	(b) Junction tree associated with the topology . . . . .	47
4.5	Tracking result for uniform and non-uniform chained-form regions . . . . .	49
	(a) Uniform distributed corridor . . . . .	49
	(b) Non-uniform distributed corridor . . . . .	49
4.6	Tracking result in random distributed networks and associated clique tree . . . . .	50



(a)	Tracking result with low noise level, $\sigma = 0.1$ . . . . .	50
(b)	Tracking result with high noise level, $\sigma = 0.5$ . . . . .	50
4.7	Estimation error for random distributed sensor network . . . . .	51
(a)	Estimation error with low noise level, $\sigma = 0.1$ . . . . .	51
(b)	Estimation error with high noise level, $\sigma = 0.5$ . . . . .	51
5.1	Graph representation and Markov random field representation of the same sensor field. . . . .	58
(a)	Network topology with all possible edges shown . . . . .	58
(b)	Network topology after assuming Markov property . . . . .	58
5.2	Similar representation of various sized targets in the sensor network. . . . .	59
(a)	Representation of a small target in sensor networks . . . . .	59
(b)	Representation of a large target in sensor network . . . . .	59
5.3	Boundary lines in regularly distributed network . . . . .	60
5.4	Converting the randomly distributed sensor network into a grid-like clique structure . . . . .	63
(a)	Random deployed sensor field . . . . .	63
(b)	Grid-like structured clique topology . . . . .	63
5.5	Converting the randomly distributed sensor network into a grid-like clique structure . . . . .	64
(a)	Triangulated sensor field with coverage holes . . . . .	64
(b)	Virtual triangles constructed to get across void areas . . . . .	64
5.6	Large target in a grid sensor field using Ising model . . . . .	68
5.7	Large target in a randomly deployed sensor field . . . . .	69
5.8	Detection of two large targets in the sensor network . . . . .	70
5.9	Small target trajectory in the sensor field . . . . .	71
5.10	Number of Messages sent by each method . . . . .	72
6.1	The estimation process of each sensor in a hallway, given the trainer location	78
6.2	Tetrahedron formed with the sensor and the triangular trainer . . . . .	79
6.3	Alternative switch pattern of hallway sensors . . . . .	81
6.4	Target oriented sensor switch pattern in hallway . . . . .	82
6.5	Target oriented sensor switch pattern in randomly distributed network topology . . . . .	83
6.6	Sensor switching pattern in hallway after the active sensor is identified . . . . .	84
6.7	Snapshot of self-localization in hallway . . . . .	85
6.8	Snapshot of self-localization in an open space . . . . .	85
6.9	Estimation error for sensor locations . . . . .	86
7.1	System Architecture . . . . .	91
7.2	Bumblebee radar used in this chapter . . . . .	92
7.3	MSP 430 sensor board used in this chapter . . . . .	92
7.4	Possible state of the elder's activity . . . . .	93

7.5	Possible state of the elder's activity . . . . .	94
7.6	Hidden Markov model for elder's activity . . . . .	95
7.7	Speed profile for a falling scenario . . . . .	100
7.8	State probability for falling and walking . . . . .	101
7.9	Speed profile for a walking to standing scenario . . . . .	102
7.10	State probability for falling and walking . . . . .	102
7.11	Functionality flow chart of the fusion center . . . . .	103

# List of Tables

5.1 Number of messages sent by each method . . . . . 71

## Preface

This dissertation contains my research work performed during the pursuing of the Ph.D. degree in Computer Engineering at Michigan Technological University. The major contributions of this dissertation is to develop a low cost, energy efficient algorithm for multiple target tracking in an unknown sensor network with random distributed network topology. This dissertation includes previously published articles in Chapter 3, Chapter 4, and Chapter 5.

Chapter 3 contains portions of one articles previously published by IEEE, and another article that is current under review. As the first author, I identified the research issue which was to detect and track mobile target in a chained-form network using dynamic clustering and Viterbi algorithm. With the guidance from the second author, Dr. Zhijun Zhao, and my advisor, Dr. Jindong Tan (the third author), the algorithm design, simulation, and setup are performed by me at the Robotics lab in Michigan Technological University. The articles were completed by Dr. Zhijun Zhao, Dr. Jindong Tan and me.

Chapter 4 contains also portions of one article previously published by IEEE. As the first author, I identified the research issue which was to broaden the scope of the chained-form sensor network to a 2-D sensor field with random network topology. With the guidance of my advisor (the second author of the article), I completed the algorithm design, simulation, and experiment in the Robotics lab in Michigan Technological University. The article was completed by my advisor, Dr. Jindong Tan and me.

Chapter 5 is again from one article previously published by IEEE, and a journal version is currently being reviewed for a transaction. As the first author, I identified the research issue which was to use conditional independency and junction tree method to convert the random network topology into a more manageable grid-like topology, and then I introduced a technical that was used in image processing to perform target tracking in the grid-like topology. With the guidance of my advisor (the second author of the article), I completed the algorithm design, simulation, and experiment in the Robotics lab in Michigan Technological University. The article was completed by my advisor, Dr. Jindong Tan and me.

## Acknowledgments

First and foremost, I would like to express my gratitude to my advisor Dr. Jindong Tan for his guidance and support on my study and research at Michigan Technology University. His expertise in distributive system and parallel signal processing intrigued me deeply into the world of robotics and wireless sensor networks. His guidance helped me getting through all the difficult times in my research and writing of this thesis. I could not have wished having a better advisor for my Ph.D research.

I would also like to thank the rest of my committee members, Dr. Zhijun Zhao, Dr. Jeffrey Burl, and Dr. Nilufer Onder for taking the time to review my dissertation, provide encouragement, insightful comments, and ask excellent questions during the actual defense.

Let me not forget to thank all my labmates, who have added much excitement to my life at Michigan Tech, I couldn't even think about what life would be like at Houghton if they weren't there to keep me entertained. They are Huaming Li, Shuo Huang, Sheng Hu, Fanyu Kong, Xi Chen, Xinying Zheng, Ya Tian, Xiaolong Liu, Zhenzhou Shao and Jian Lu. They have been great collaborators during study, and great playmate at sparetime.

Last but not the least, I would like to thank my wife, Ying Li, for taking great care of me so that I can focus on my thesis, and thank my parents, Shangzhao Shi, Xiuling Li to provide support and encouragement for all my life. I could not have gone this far without them.

## Abstract

Sensor networks have been an active research area in the past decade due to the variety of their applications. Many research studies have been conducted to solve the problems underlying the middleware services of sensor networks, such as self-deployment, self-localization, and synchronization. With the provided middleware services, sensor networks have grown into a mature technology to be used as a detection and surveillance paradigm for many real-world applications.

The individual sensors are small in size. Thus, they can be deployed in areas with limited space to make unobstructed measurements in locations where the traditional centralized systems would have trouble to reach. However, there are a few physical limitations to sensor networks, which can prevent sensors from performing at their maximum potential. Individual sensors have limited power supply, the wireless band can get very cluttered when multiple sensors try to transmit at the same time. Furthermore, the individual sensors have limited communication range, so the network may not have a 1-hop communication topology and routing can be a problem in many cases.

Carefully designed algorithms can alleviate the physical limitations of sensor networks, and allow them to be utilized to their full potential. Graphical models are an intuitive choice for designing sensor network algorithms. This thesis focuses on a classic application in sensor networks, detecting and tracking of targets. It develops feasible inference techniques for sensor networks using statistical graphical model inference, binary sensor detection, events isolation and dynamic clustering. The main strategy is to use only binary data for rough global inferences, and then dynamically form small scale clusters around the target for detailed computations. This framework is then extended to network topology manipulation, so that the framework developed can be applied to tracking in different network topology settings.

Finally the system was tested in both simulation and real-world environments. The simulations were performed on various network topologies, from regularly distributed networks to randomly distributed networks. The results show that the algorithm performs well in randomly distributed networks, and hence requires minimum deployment effort. The experiments were carried out in both corridor and open space settings. A in-home falling detection system was simulated with real-world settings, it was setup with 30 bumblebee radars and 30 ultrasonic sensors driven by TI EZ430-RF2500 boards scanning a typical 800 sqft apartment. Bumblebee radars are calibrated to detect the falling of human body, and the two-tier tracking algorithm is used on the ultrasonic sensors to track the location of the elderly people.

# Chapter 1

## Introduction

A wireless sensor network is a distributed system which consists of large amounts of small and inexpensive sensor nodes. Each sensor node in the network has the ability to observe the surroundings and to process the observed data, hence providing a method to interact with the physical world. However, individual sensors are usually unreliable, have limited sensing resolution, and have limited processing capability. Therefore, individual sensor readings cannot accurately reflect the events in the physical world. With the wireless communication capability, the sensor nodes can autonomously form an ad hoc network, where the data from individual sensors can be aggregated, and eventually recovering the details of the events in the physical world. Because there are a large number of sensors in the network, the impact of individual sensor failures and miss detections are automatically minimized due to redundant data from the neighbors. The small physical size and wireless communication abilities allow the sensors to be placed around obstacles, and to be placed in space constrained places to provide direct line-of-sight measurements. In short, wireless sensor networks offer a distributed, flexible, non-intrusive platform for observing the physical world.

However, there are a few physical limitations to the sensor network itself, which can prevent sensors from performing at their maximum potential. Individual sensors have limited power supply. It is a common assumption that once the sensors are deployed it is difficult to recycle for charging or battery replacement, hence prolonged battery life is one of the major topics in sensor network research. Due to the large number of wireless sensors, the wireless band can get very cluttered when multiple sensors try to transmit at the same time. Hence Media Access Control is another major research topic in the sensor networks. The individual sensors also have limited communication range. A 1-hop communication may not be possible within the network, thus routing can be a problem in many cases. In addition to the physical limitations, in wireless sensor networks, there are multiple types

of sensors observing one or more events in the physical world from different perspectives. There will be a large amount of data streaming from different sensors. Some of these data will be correlated, and some of these data will be independent, and some of the data may be observed from a different event. Hence, inferences using graphical models are needed to fully understand what the sensors are referring to in the physical world. An inference is a prediction made by using the statistical and heuristic techniques. It can be as simple as interpolating the temperature reading between two sensors based on the distance between them. Or, it can also be very complex as to identify a suspect from a busy street based on observation. In this thesis, inference techniques for target tracking with wireless sensor networks are developed.

Target tracking has a wide range of civilian and military applications. It is widely used in traffic control, surveillance, emergency response systems, search and rescue operations, supply chain management, and battle field awareness systems. Utilizing wireless sensor network for target tracking is especially beneficial as the sensor network is non-intrusive and requires less infrastructure. However, the individual sensors in a sensor network have limited sensing and computation capabilities. Hence, tracking of a mobile target cannot be done by individual sensors, sensors must operate collaboratively. To collaborate, information must be shared among sensor nodes, but sensor nodes also have limited communication bandwidth, hence a comprehensive system is needed to balance local processing and data propagation.

## **1.1 Common Inference Method Used in Wireless Sensor Networks**

The main focus of this thesis is to perform distributed processing for inference and estimation in randomly deployed sensor networks with the aid of graphical models. To analyze this problem, we generally utilize two tools, the Bayesian framework and a graphical model for topology manipulation. The Bayesian framework is one of the most popular choice for inferences and estimation, where the Bayes rule models the truth state and evidence in a single equation. Graphical models, especially Markov random fields, cliques and junction trees are used to organize randomly distributed networks into more manageable network topologies for ease of processing and communication.

Bayesian framework is one of the most classical estimation methods. It utilizes Bayesian rule to combine true state of the object and evidence in a single equation, hence provides a posterior distribution to describe the probability of possible true states. When Bayesian framework is applied to tracking, the states are linked by a temporal relation. The current



state can be predicted from the state in the last time frame, combined with the observation from current time frame, and normalized to obtain the a posterior distribution for the current state. For tracking applications with Bayesian framework, a temporal Markov property is usually assumed, hence the current state is only dependent on the state immediately before it, but not all of the previous states.

To use Bayesian framework efficiently, the required information such as new observations and previous state distributions must be provided to the estimator. However, in typical sensor networks with many sensors, only a few of them are actually needed to estimate the location of a target. Feeding every bit of information from every sensor in the network to the Bayesian framework for collective computation is not only time consuming, but it also dissipates a large amount of unnecessary sensor energy. A graphical model is then used to isolate the target with the Markov property, cluster the sensors into small group with cliques and junction trees. Then, the computation can be applied to only a small cluster of sensors instead of the entire sensor field.

## **1.2 Main Challenges of Tracking and Detection in Wireless Sensor Network**

This thesis focuses on the challenge of performing accurate tracking and estimation using only a small amount of resources, and managing the network topology in a dynamic way using graphical models. In particular, we consider three sub-problems that contribute to the main challenge.

- † Two-tier target tracking framework, with regular and irregular network topology
- † Ultrasonic sensor network self-localization and scheduling
- † Falling detection in elderly homes using wireless sensor networks

Each chapter in this thesis is devoted to one of the sub-problems, and more in-depth exploration of the problem itself and used techniques are presented in each chapter.

## 1.3 Thesis Organization

For each of the sub-problem listed, we explore the theory and sources behind the difficulties, quantify the difficulties, derive equations and propose algorithms to solve the difficulties. Simulations and experiments are performed to verify the proposed algorithms. The efficiency and accuracy of the algorithms are measured and presented. The following sections of the thesis are organized as follows.

**Background.** The relate literature is reviewed briefly in Chapter 2. The review starts out with the basics of the Bayesian framework, followed by tracking algorithm using the Bayesian framework. The strength and shortcomings of the algorithms for sensor network applications are discussed. The graphical model approaches and some random sampling techniques are then discussed in this chapter, their advantages in topology manipulation are described.

**Two-tier Target Tracking Framework** The two-tier tracking framework is discussed in chapter 3 [1]. Tracking in wireless sensor networks is different from the traditional tracking with a centralized system. A wireless sensor network is a distributed platform, where the topology of the network is an important aspect that needs to be considered. For sensor network deployed in real environments, the network topology is unknown in the beginning, and is subject to quite frequent changes due to sensor displacement or sensor failure. The common approach to deal with an uncertain topology is to utilize a statistical graphical model to solve the tracking problem in the sensor networks. This maps the target tracking problem in sensor network into a statistical inference problem in graphical model. A statistical graphical model handles the uncertainty in the sensor network by express the sensor nodes with random variables, and the interconnection between sensors with correlations. The graphical model also provide a framework for the message passing technique, allowing data to be fused and disseminated efficiently.

In this thesis, we model target tracking using Bayesian framework, and emphasize the distributed topology of the network using graphical model. A framework for a two-tier target tracking algorithm in an arbitrary network topology is presented. In tier-one, binary sensor data is used to quickly determine a rough location of the target, and in tier-two, only a small number of sensors are used to compute the precise location of the target. By doing so, the tracking algorithm enjoys both the precision of detailed local inference, and the simplicity of having abstracted information of the entire network. An ultrasonic sensor network was used to implement the tracking framework in a real-world hallway-room setting.

**Tracking in sensor network using clique tree approach** Chapter 4 serves as an extension

to the two tier target tracking framework described in Chapter 3. In addition to dynamically clustering the sensors into local inference groups, it also constructs a regularly distributed clique structure from the irregularly distributed sensor fields using cliques in a Markov random field. By having a regularly distributed clique network, the clustering of the sensors becomes less ambiguous, hence greatly improves the performance of the sensor clustering [2].

The algorithm developed in this chapter triangulates the sensor network under the guidance of a Gabriel Graph (GG). Each triangle is combined into a super-node, and each super-node can have exactly 3 connected neighbors. Hence a tree structure can be constructed, with the starting location of the target as the root, and each node down the tree has exactly 3 children. With this approach a simplified network structure is formed to apply the two-tier framework.

**Large Target Tracking in Randomly Distributed Network** Chapter 5 extends the work in Chapter 4, where the clique tree approach is taken to a new perspective. We apply the concept of a Markov random field into sensor networks by identifying the conditional independency of the data between non-adjacent sensors given the sensors in the middle. Once this independency is identified, the sensors can be naturally grouped into independent cliques, and the sensors shared between clusters become links that ensure data propagating through the network. After the individual sensors are combined as cliques, the topology of the network will be greatly simplified, hence makes the two tier target tracking much easier to apply. In this system, a large area of effect targets can be identified and tracked, since the event boundary can be easily identified by isolation of the affected sensors [3].

**Ultrasonic Sensor Network Self-localization and Scheduling** Chapter 6 primarily deals with the physical implementation of the challenges of the tracking framework. An ultrasonic sensor network is implemented to conduct the target tracking. However, in the physical implementation, we expect the sensors to be able to localize themselves, form the ad hoc network autonomously, and to minimize the ultrasonic interference. Hence, a sit-in scheme is needed for the sensor network before they can be used for tracking purposes.

This chapter focuses on the challenges of localization, energy, coverage and interference management of the ultrasonic sensor networks. A feasible localization technique for randomly distributed ultrasonic sensor networks is provided. Then a scheduling technique is discussed to optimize the energy consumption, enlarge the sensor coverage region, and minimize the ultrasonic over hearing and interference. Since our application for the ultrasonic sensor network is tracking and localization of mobile target, the sensor scheduling technique is presented in two difference scenarios, when targets are presenting, and when no target is presenting. It is shown that the target will help sensor scheduling even if the target is non-cooperative.

**Falling Detection in Elderly Homes** A practical application for target tracking framework is considered in Chapter 7. With the aid of an on board bumblebee radar sensor, the ultrasonic sensor network is able to predict, detect, and alert falling incidents happening in elderly homes. Falling is one of major causes of severe injuries and death of the elderly people who are living independently. It is estimated that more than 30 percent of people over the age of 75 experience a falling incident each year. It is desirable to have a system that reports the real-time status of elderly person, and to automatically call for assistance when falling incidents occur. There are several existing systems that allow elders to call for emergency assistance by pushing a button on a wearable device. However, in many cases, the falling will result in unconsciousness, hence the person will not be able to push the button. Other pre-existing conditions such as dementia also prevent the elders to be able to operate the device. A falling detection system is desired, which can efficiently determine when falling occurs and when assistance is needed.

In this chapter, an ultrasonic sensor network based falling detection system is presented. The system is able to detect falling people using ultrasonic ranging and speed measurement obtained through an onboard radar system. In addition to falling detection, the system is able to provide location information and the elder's movement pattern, which can be used to estimate the likelihood of falling ahead of time in order to efficiently prevent the falling from happening. The networks are connected through a base-station node that forwards the timely data to the computer, then transmitted to the appropriate personnel. The hardware platform presented in this thesis is an ultrasonic sensor network with small scale radar node for speed detection. The system itself is integrated to the house infrastructure, hence no wearable device or dependence on the elderly person is required.

# Chapter 2

## Background

In this section, a general overview of related theory, algorithms, and methodology is provided. These are tools that are used throughout the rest of this thesis. The general concept of wireless sensor networks is introduced the first, and then inference in sensor networks is discussed. Bayesian framework is presented the next, followed by its application in tracking problems, and its variations such as binary data methods, random sampling methods. Graphical model based method is introduced in the end, we especially focused on clique and junction tree concept, and then we present some exist tracking algorithms that take the network topology into consideration.

### 2.1 Applications and Benefits of Wireless Sensor Networks

Wireless sensor network has become popular choices for many civilian and military applications [4] [5] [6] [7] [8]. Wireless sensor networks are constructing from many inexpensive sensor nodes, which are capable of sensing and perform light computation independently. When many of these sensor nodes connect together wirelessly to form an ad hoc network, they could tackle on more difficult problems via collaboration. Due to the large physical area the sensor network could cover, sensor network is especially suitable for detection, tracking and navigation applications.

Compared to traditional centralized sensing and detection methods, a distributive sensor network can provide several benefits. In sensor networks, individual sensor nodes are small in size, by using the wireless communication capability, they can be deployed into various

locations with space constraints, hence providing the direct line of sight. Since a collective decision is made among sensors using statistics, individual sensor failures affect little on the final results [9] [10].

The sensor network also has several limitations. Sensor nodes are powered off self-contained power source (i.e. batteries). After deployment, recharge or replace the battery is usually infeasible. Hence, applications designed for sensor networks must keep the energy consumption in mind [11] [12]. Sensors are equipped with wireless communication capability, in an infrastructure-less ad hoc network setting, contention for the channel, arranging data transfer based on importance can be a problem. The microprocessor in the sensor node is sufficient for applications ranging from simple data compression to complex distributive algorithms, however, the computation power is still limited by the memory footprint and data storage space. When developing algorithms in this thesis, all of these limitations are taken into consideration.

## **2.2 Fundamental Inference Engine in Wireless Sensor Networks**

The Bayesian framework is the fundamental method for solving many detection and estimation problems. Since the wireless sensor network is specialized in detection and tracking applications, the Bayesian framework also serves as the fundamental inference engine in wireless sensor networks.

### **2.2.1 Bayesian Framework Formulation**

The Bayesian framework is widely used to solve detection and estimation problems. It is based on the famous conditional probability equation, the Bayes rule.

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \quad (2.1)$$

In detection and estimation problems,  $X$  is modeled as the true state of the target, and  $Y$  is modeled as the observation.  $P(X|Y)$  is a posterior distribution describes the probability

of having state  $X$  given observation  $Y$ . In order to compute this posterior distribution,  $P(Y|X)$ ,  $P(X)$  and  $P(Y)$  need to be defined.  $P(X)$  is a prior about the target state, which is predetermined by empirical data, and  $P(Y)$  is the prior of observation, and it is usually used as a normalization constant.  $P(Y|X)$  is the observation model, usually given in form of a state equation.

The general Bayesian framework applied to tracking problems can be expressed as,

$$\begin{aligned} P(X_t|Y_{1:t}) &= KP(Y_t|X_t)P(X_t|Y_{t-1:t}) \\ &= KP(Y_t|X_t) \int P(X_t|X_{t-1}, Y_{1:t-1})P(X_{t-1}|Y_{1:t-1})dX_{t-1}. \end{aligned} \quad (2.2)$$

In this equation,  $P(Y_t|X_t)$  is the observation model,  $P(X_t|Y_{t-1:t})$  is the prior of  $X_t$ , which is a prediction of possible  $X_t$  given all of the previous states [13].

### 2.2.2 Bayesian Framework Based Tracking Algorithm

The Multiple Hypothesis Tracking (MHT) is one of the most fundamental tracking algorithms in sensor networks [14]. It is based on the Bayesian filtering framework, which exhaustively enumerate possible target actions and the associated probability for each time frame, and then decide the best series of actions based on the probability. However, this approach suffers from hypothesis explosion, since the possible action of the target increase exponentially over time. The Joint Probabilistic Data Association (JPDA) algorithm is another Bayesian filter based algorithm [15]. In the JPDA algorithm, the possible target actions are again enumerated for each time frame, but an action will be selected before making any further hypotheses, hence greatly reduces the number of hypotheses in the future. However, this approach loses some accuracy as it may prune away feasible hypotheses.

Binary sensors can be used to reduce the communication cost and computation cost of the Bayesian filter based algorithms. The algorithm proposed in [16] is a typical example. The algorithm computes the probability of the target locating in its vicinity based on previous target location, its own observation and the observation of its neighboring sensors. The observations are only passed between neighbors, and the information passed is only a single bit. This algorithm greatly reduces the communication and computation cost, but the accuracy quite low. The sensor can only tell whether the target is in its vicinity or not. Modifications of the binary sensors approach are made in [17] [18]. All of these

algorithms tried to increase the accuracy by study the overlapping sensing region. The sensing region of each sensor is segmented into several sub-parts based on the overlap with different neighbors, the accuracy is increased to the size of the sub-region rather than the entire sensing region. However, unless in a network with uniform topology, the sub-regions are difficult to derive.

The Particle filter is another way to reduce the cost of Bayesian filter based algorithms. Although particle filter requires complete data from all sensors, however, it compresses the collected probability distributions into a single distribution with fixed number of samples (particle). Despite the collected information increases, the number of particles is always fixed, hence, the communication and computation cost is fixed. The detailed analysis of the estimation and accuracy can be found in [19]. Many variation of particle filter algorithms are developed, in [20], localized particle filter algorithm is developed. The algorithm group sensors into disjoint cliques, and standard particle filter algorithm is carried out within each clique simultaneously. The algorithms proposed in [21] further improved the localized particle filter algorithm developed in [20], where the data shared between cliques are also compressed and estimated by a Gaussian Mixture Model (GMM). Particle filter algorithm for multiple target tracking is further simplified in [22], where a control structure is developed to control the number of particles as the number of target increase in the network.

When multiple targets present, data association problem is used to associate the measurements to the corresponding target trajectory. Typical Bayesian filter approaches require a known number of targets, and the initial state of the target in order to start the association iteration. Markov Chain Monte Carlo Data Association (MCMCDA) algorithms solved this problem. Framework of MCMCDA algorithm can be found in [23] [24], where the target trajectories (tracks) are modeled as Markov Chain, and random sampling is used rather than iterative Bayesian filter, hence relaxed the dependency of the prior information.

## **2.3 Target Tracking in Sensor Networks Using Statistical Graphical Model**

Statistical graphical model combines message passing in graphs with probability. The graph captures the topology of the sensor network while message passing establish the data propagation pattern within the network. The probability links the measurements with correlation and isolates the events with independency within the sensor networks.



### **2.3.1 Graph-based Tracking Algorithms**

There aren't many graph-based tracking algorithms developed, and most of these graph-based tracking approaches nowadays are focused on topology to physical location manipulation. In [25], for example, the rooms and the hallways are modeled as nodes in the graph, the doors and passages are modeled as links in the graph, and the target is assumed to transit from node to node. The actual tracking algorithm is still under Bayesian framework, and the graph part of the algorithm is to maintain the sensors (RFID readers) to be linked in such a topology.

Another graph-based tracking method is shown in [16]. This tracking method is another distributive implementation of the Bayesian framework. In this algorithm, binary sensor data is used, and the topology information is only used to provide neighboring information and information routing paths. Some other graph-based methods study the overlapping of the sensing region with the aid of the known network topology [17], [26], [18]. All of these graph based algorithms depend heavily on knowledge of the network topology. The topology has to be regularly distributed and the node locations have to be known.

### **2.3.2 Graph-based Event Boundary Detection Algorithms**

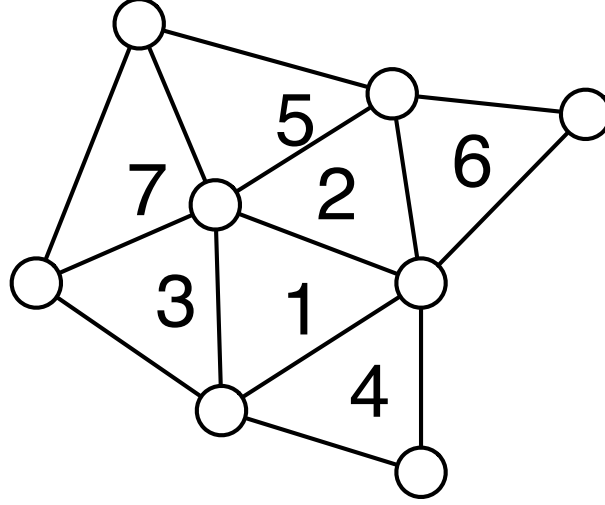
There are some algorithms developed for detecting event boundary in sensor networks that are also graph based. Differ from a single point target, an event is usually an area of interests, and tracking of such area is a boundary detection process. The algorithm proposed in [27] and [28] used a threshold-based system for each sensor to make detection decisions. The algorithm developed in [29] uses the k-nearest neighbor to group the detection sensors into clusters, and the cluster would represent an area of event. The outlier sensors are determined to be false alarms. A more advanced method is offered in [30], where the sensor field is recursively divided into sub-regions, until each sub-region contains only null detection sensors, or detection sensors. The boundary can be easily identified once these sub-regions are constructed.

### **2.3.3 Cliques, Junction Trees and Markov Network**

Although these tracking algorithms are graph-based, they did not fully utilize the data fusion, data correlation and clustering features offered by graphical model. In order to fully

utilize inter-sensor correlation, cliques and junction tree algorithms should be considered.

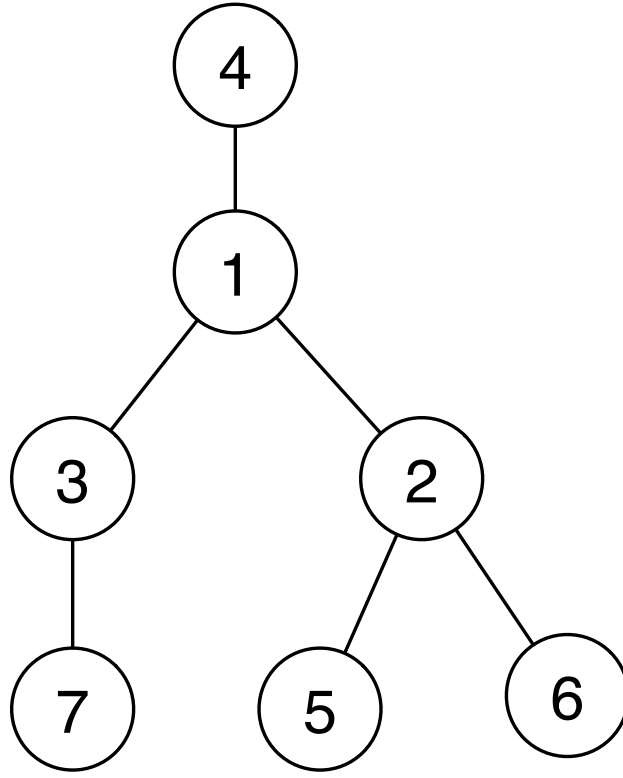
In graphical model, a clique is a group of fully inter-connected nodes. it is usually used to subdivide the graph into subgraphs. The cliques are independent to each other given the shared nodes, and the statistical property of the clique is described in terms of clique potentials, which is usually the joint probability of all the enclosed node.



**Figure 2.1:** Illustration of a clique

Clique tree is a tree that all of the nodes are cliques. It provides an overview of the graph after the graph is been subdivided using cliques [31] [32]. Junction tree is a special case of clique tree that obeys the running intersection property [33] [34]. The running intersection property basic dictate the locality property in the clique tree, it requires that for any pair of cliques  $U$  and  $V$ , the cliques between  $U$  and  $V$  must contain  $U \cap V$ . While the cliques provide clustering and joint probability of a small group of nodes, the junction tree provide a fast route to propagate information through the network, and it is also statistically sufficient to represent the original graph.

Markov network is a special type of graph, where the edges correspond to direct probabilistic interaction between neighboring nodes, and is not to be intervened by other nodes, e.g. the Markov property [35] [36]. It is commonly used for segmentation in image processing [37] [38] [39] [40]. For tracking and detection applications in sensor networks, the target is usually concerned by a small set of local sensors, hence Markov network is especially suitable for such situation. Combine Markov network, junction tree and clique concept, a randomly deployed sensor network can be converted into a more structured form,



**Figure 2.2:** Illustration of a junction tree

and greatly simplify the collective inference process and data propagation.

## 2.4 Physical System of Our Ultrasonic Sensor Network

We developed custom ultrasonic sensors to setup the test bed for our experiments. The ultrasonic sensor is interfaced with MSP430 microprocessor and CC2500 RF transceiver for data processing and information propagation. In this chapter, the ultrasonic sensor will be presented, the sensors are calibrated, and calibration result is shown. Then the localization and synchronization techniques used in the experiments are discussed.

### **2.4.1 Physical Properties of The Ultrasonic Sensor Nodes**

The ultrasonic sensor used in my experiments is LV-MaxSonar-EZ0 sensor from MaxBotix. The sensor range is 6 to 254 inches with a high resolution of 1 inch. The ultrasonic transducer is operated at 25 kHz frequency. The sensor board can be operated with 2.5V to 5.5V supply voltage, and can obtain a distance reading every 49 ms.

The ultrasonic sensor provides three types of interfacing method. A RS232 pin which provide serial data at baud rate of 9600. An analog pin with an amplitude reading, where a reading of  $(V_{cc}/512)$  mV/inch can be obtained. A pulse width modulation pin with  $147\mu s$  represent 1 inch of distance. In my experiences, I am planning to interface the ultrasonic sensor board with MSP430 microcontroller through the pulse width modulation pin. The MSP430 will perform a timer capture function to capture the width of the pulse, hence compute the distance.

### **2.4.2 Self Localization of The Ultrasonic Sensor Network**

One of the main contributions in this thesis is to be able to utilize sensor network with random network topology for tracking, hence minimal deployment effort is required. The initial positions of the sensors are unknown. The sensors need to perform self-localization before they can do actual tracking. Due to the lack of GPS device and anchor nodes, the absolute position of the sensors cannot be determined. With distance measurement between sensors, a relative position between sensors can be obtained.

Since sensors are equipped with ultrasonic sensors, they can measure the distance to any of the neighbors. A triangle can be obtained if three distance measure can be obtained among three sensors (A triangle can be constructed with three known sides). Additional sensors can be added to form a chain of triangles. Furthermore, in our algorithm, a Gabriel Graph (GG) is constructed. With only the distance information, the GG graph can be constructed.

### **2.4.3 Timer Difference of Arrival Approach**

Each sensor node is equipped with ultrasonic sensor, so technically it should be simple to compute the distance between the sensors. However, the sensors must be synchronized in order to utilize the ultrasonic sensor effectively, because the sensor must know the time

interval between transmitting and receiving. If the transmitter and receiver nodes are not synchronized, this time interval cannot be measured.

In our ultrasonic sensor network, however, It is possible to measure distance between two sensor nodes without synchronization. In addition of the ultrasonic sensor, each node is also equipped with RF radio. If we send a radio signal at the same time as the ultrasonic signal, by capture the time difference of arrival, we can estimate the distance between sensors by solving the following equation.

$$T_{diff} = \frac{D}{v_{sound}} - \frac{D}{v_{light}}, \quad (2.3)$$

where  $T_{diff}$  is the time difference between arrival,  $D$  is the distance between sensors,  $v_{sound}$  is the speed of sound and  $v_{light}$  is the speed of light.

## 2.5 Synchronization Techniques in Sensor Networks

Although synchronization is not needed to construct the GG graph, it is still necessary for the sensors to be synchronized for tracking applications. There are many well developed synchronization algorithms, we will simply pick one of them to be used in our experiments.

### 2.5.1 Reference Broadcasting Synchronization (RBS)

The Reference Broadcasting Synchronization (RBS) is one of the most accurate synchronization techniques [41]. It is simple to implement, and provide high accuracy synchronization. In RBS, a reference sensor is selected, and the reference sensor broadcast a periodic beacon message. All sensors receiving the beacon message adjust the clock according to it. The advantage of such synchronization technique is that it cuts the critical path in half, the error from the transmitter side is completely eliminated, and only the error from the receiver side is included. This tracking algorithm has an extreme small footprint in the ram, and is not likely to incur errors during execution. However, the maintenance cost of this approach can be high, as it requires pre-installed reference sensors, and they need to keep broadcasting while the operation lasts.

### 2.5.2 Romer's Time Synchronization Method

Another type of time synchronization method is developed by Romer etc [42]. In his time synchronization method, instead of aligning the clock of the sensors via sync packet, each clock keeps on its own pace, and for each transmission, a very accurate propagation delay is estimated. The idea is upon sending in message, the sender would put a time stamp  $T1$  onto the packet based on sender's clock. Upon receiving the packet, another stamp  $T2$  is generated based on receivers clock. The receiver then sends back an ACK message, with timestamp  $T3$  based on its clock, and upon receiving ACK, sender would generate time stamp  $T4$  based on sender's clock. Once the four time stamp is generated, the propagation delay can be estimated as  $((T2 - T1) + (T4 - T3))/2$ .  $T2 - T1$  is essentially the clock difference plus the propagation delay, while  $T4 - T3$  is the negative of the clock difference, plus the reverse propagation delay. When adding together, the clock difference cancels each other, while the propagation doubles, hence dividing by 2 would grant us a very accurate propagation delay. Since propagation delay is a major part in the critical path of time synchronization, and this value is subject to change with the network condition, this algorithm performs very well at dynamically finding propagation delay at all time. However, this does add on to the cost of each message transmitted, and adds overhead to each calculation.

### 2.5.3 Post Facto Synchronization

This is yet another type of unique time synchronization techniques that is being utilized in sensor networks. The idea behind this algorithm is that the sensor network remains unsynced, until an event occurs, and adjacent sensor would use the detected events as a reference, and go back in time to synchronize their clocks with each other. This approach requires practically no maintenance cost. However it can add large delays and overheads when events do occur.

# Chapter 3

## Two-tier Tracking System <sup>1</sup>

### 3.1 Introduction

There are many challenges in mobile target tracking in distributed sensor networks. Resource limitation in wireless sensor network is one of the main challenges, the sensors have limited sensing, computation and communication capability, and hence individual sensors cannot track the targets alone. Collaboration is needed to accurately locate the targets. However, excessive collaboration induces heavy amount of communication, where the communication power is also very limited, hence balance between local processing and collaboration need to be determined. Wireless sensor network is full of uncertainties. Individual sensors usually have low sensing resolution, and they are prone to failures, hence in-network signal processing is needed to process the data collected from the sensors before any decisions can be made [26]. For target tracking applications, a real-time report of the target location is required, but when the networks size is large, it is time consuming to just gather information through the entire network, hence local processing is required to produce real-time tracking results [18]. Finally multiple targets have always been problematic, even before the trajectories cross, confusions can arise when combine data that are collected from different targets. When targets trajectories cross, new target appears, and target disappears, the ambiguity grows higher [14] [15].

A two-tier graphical model based tracking algorithm is developed in this chapter. The

---

<sup>1</sup>©2007 IEEE. Portions reprinted with permission, from **Lufeng Shi**, Zhijun Zhao, and Jindong Tan, “*Near optimal two-tier target tracking in sensor networks*”, in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1993 - 1996, 2007. See Appendix for a copy of the copyright permission from IEEE.

algorithm assumes a known sensor network topology and Markov transition between target positions. The first tier of the algorithm is a detection algorithm, it utilize the Viterbi algorithm to compute a rough target location. In this tier, only a single bit of information is transmitted, this bit will indicate the likeliness of having a detection event in the vicinity of the sensor. In the end of this tier, one sensor will confirm the detection event in its vicinity, and the location of this sensor is assigned to the rough target location. With the rough target location known, a cluster is formed around this rough location with a small number of sensors in the second tier. The sensor confirmed the detection event in the first tier will serve as the cluster head. A detailed evaluation of the target location will be carried out within this small cluster. With this two-tier algorithm, we avoid to generate and circulate large amount of data around the entire network, while kept very good accuracy by performing detailed measurement and computation in small scale.

## 3.2 Related Works

Mobile target tracking has been a classic research topic, and sensor networks have brought new vitality to it. The tracking framework for multiple targets are provided by Multiple Hypothesis Tracking (MHT) [14]. It exhaustively enumerates possible target actions and the associated probability for each time frame, and then decides the best series of actions based on the probability. However, this approach suffers from hypothesis explosion, since the possible action of the target increase exponentially over time. Joint Probabilistic Data Association (JPDA) made these hypotheses based data association more feasible [15]. In JPDA algorithm, the possible target actions are enumerated and compressed into a single posterior distribution, the decision is made by maximizing the posterior distribution at each time frame, hence greatly reduces the number of hypotheses in the future.

Although these frameworks provide theoretic background for tracking in sensor networks, however, the computation limitation of sensors, and propagation of the distributed data are not considered. In order to implement feasible tracking algorithms in sensor networks, data routing, data aggregation, small data size, and distributed computing must be considered. Researchers generally took two directions in solving such problem. The first direction is binary sensor data and the second approach is Monte Carlo sampling.

Binary sensors are used to reduce the communication cost and computation cost. The algorithm proposed in [16] is a typical example. The algorithm computes the probability of the target locating in its vicinity based on previous target location, its own observation and the observation of its neighboring sensors. The observations are only passed between neighbors, and the information passed is only a single bit. This algorithm greatly reduces the communication and computation cost, but the accuracy quite low. The sensor can only



tell whether the target is in its vicinity or not. Modification of the binary sensors approaches are made in [17] [18]. All of these algorithms tried to increase the accuracy by study the overlapped sensing region. The sensing region of each sensor is segmented into several sub-parts based on the overlap with different neighbors, the accuracy is increased to the size of the sub-region rather than the entire sensing region. However, the sub-regions are difficult to derive, unless in an uniform topology network.

In cases where parametric distribution cannot be derived, Monte Carlo sampling are used. Particle filter is developed to use Monte Carlo sampling to deal with cases where non-parametric distributions are obtained. Although particle filter requires complete data from all sensors, however, it compresses the collected probability distributions into a single distribution with fixed number of samples (particle). Despite the collected information increases, the number of particles is always fixed, hence, the communication and computation cost is fixed. The detailed analysis of the estimation and accuracy can be found in [19]. Many variation of particle filter algorithms are developed, in [20], localized particle filter algorithm is developed. The algorithm group sensors into disjoint cliques, and standard particle filter algorithm is carried out within each clique simultaneously. The algorithms proposed in [21] further improved the localized particle filter algorithm developed in [20], where the data shared between cliques are also compressed and estimated by a Gaussian Mixture Model (GMM).

Sensor networks have distributed network topologies, hence routing data to the relevant sensors can be a challenge. Clustering is a common approach to deal with such methods. Within each cluster, a cluster head node is usually elected, and the information are all passed to the cluster head for processing. Voronoi diagram and Delaunay triangulation is usually used to define the clusters [43] [44]. Newer development on clustering dynamically group sensors together by predicting the future target location [45] [46], only the sensors that falls on the future path of the target are clustered. Some algorithms are able to remove redundancy within the cluster [47], if the sensors in the cluster have too much redundancy, it is removed from the sensor. Other clustering methods in sensor network are typically used to confine the computation into a finite number of sensors. They are semi-centralized as a cluster head is selected, and performing the computation. The rest of the members in the network are typically just providing sensing data [48] [49] [50].

Clustered sensor network can be re-arranged into a tree shape with each cluster as a “super” node. The tree shaped structure significantly improves the data routing. STUN and DOT both utilized a pre-clustered sensor field, and constructed a spinning tree, the tracking in then carried out on the spinning tree [51] [52]. The OCO technique deals with routing problem by using distance vector to construct a shortest path table for the network [53], the data routing and aggregation is then carried out according to the table. In OCO, however, the connection table must be maintained and updated in a timely manner to deal with sensor failures.

Message passing algorithm combines data routing and data aggregation into a single step [54]. However, when loops present, the message passing algorithms would not be able to produce an exact solution. The earlier works usually cut the loop open by terminate the message passing after the same message loop through a network node for certain number of iteration, but using this method the converge time is greatly dependent on the size of the loops. The newer development in graphical models suggest to construct a junction tree on the network [55] [56] [57], and the loops are combined into individual cliques, hence the loops are removed. This approach adds additional computation cost, as it needs to construct and maintain a junction tree over the network.

There are other algorithms that utilize binary sensor for clustering and tracking similar to this thesis. An Triplet Circle Intersection method is developed in [58], which uses binary sensor to cluster three sensor node together, and then estimate the detailed target location using intersecting arcs between the three sensing region. This algorithm has more constraints, but still unable to achieve the same degree of accuracy compare to the two-tier system. This will be discussed more in the comparison section at the end of the chapter.

### 3.3 Problem Definition

Targets in a tracking problem may vary. Depends on the application, the target can be human, animals, objects, containers, vehicles, or even information packets in a computer network. In this thesis, we particularly interested tracking objects that can be modeled as a point mass. This implies that the dimension of the target should not be too large, and the average distance between sensors should be much greater than the target itself. In this case, human or mobile robots are good examples for point mass targets. The instantaneous coordinate of the target can be expressed as  $(x_t, y_t)$ .

The sensor model for individual sensors in the network is simple, each sensor are equipped with a single ultrasonic sensor, the sensor will be able to measure the distance between the sensor itself and the target, however, the bearing angle of the target is unknown. This can be expressed mathematically as,

$$m = r + n, \tag{3.1}$$

where  $m$  is the measurement at the sensor location,  $r$  is the actual distance between the sensor and the target, and  $n$  is random noise. For detection purpose, we could directly

compare this  $m$  value with a pre-selected threshold  $\tau$ ,

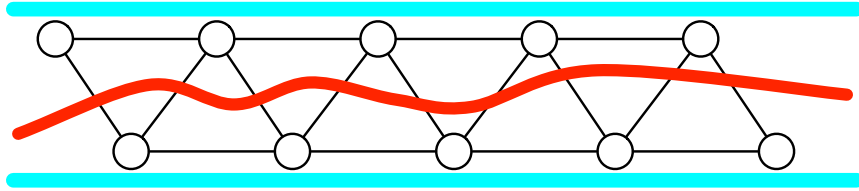
$$D_i^t = \begin{cases} H_0 & \text{if } m > \tau; \\ H_1 & \text{if } m \leq \tau. \end{cases} \quad (3.2)$$

$H_0$  implies null detection, and  $H_1$  implies detection. Since both the sensor and the target are assumed to be point mass, the actual distance between the sensor and the target  $r$  can be expressed using the coordinate of the sensor and the target,

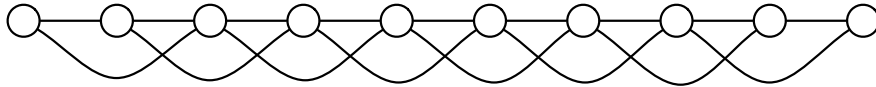
$$m = \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2} + n, \quad (3.3)$$

where,  $(x_i, y_i)$  is the coordinate of the sensor. We assume the coordinates of the sensors are known, and the target coordinate  $(x_t, y_t)$  is to be computed by the proposed algorithm.

Since we need to perform fast detection estimation over the entire sensor network in the first tier of our algorithm, a well defined topology of the network is crucial. In this thesis, we will consider three types of the network topology. The first type is a Chained-form network, as shown in Fig. 3.1. The target has the limited freedom on the side, and unlimited freedom in the forward and backward direction. This is a simplified situation, in the topology perspective, there is only a single dimensional of networked sensors chained together with 2nd order of connectivity, as shown in Fig 3.2.



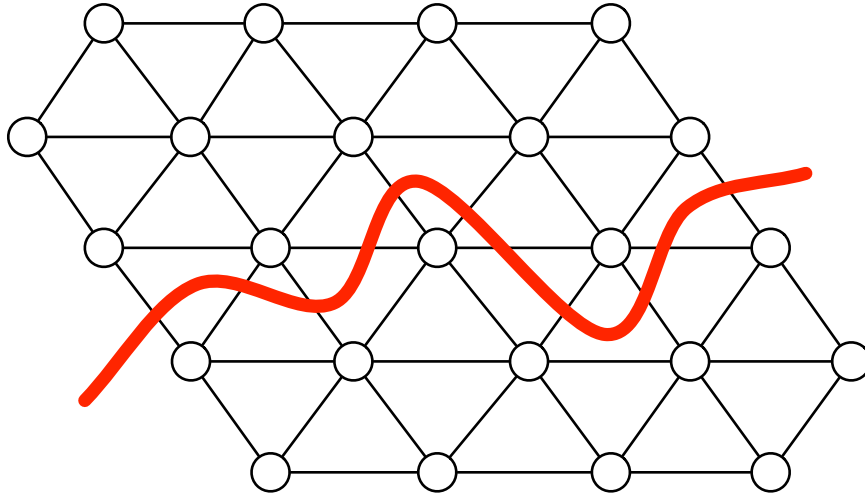
**Figure 3.1:** Chained-form network topology



**Figure 3.2:** One dimensional network topology with 2nd order connectivity derived from chained-form network

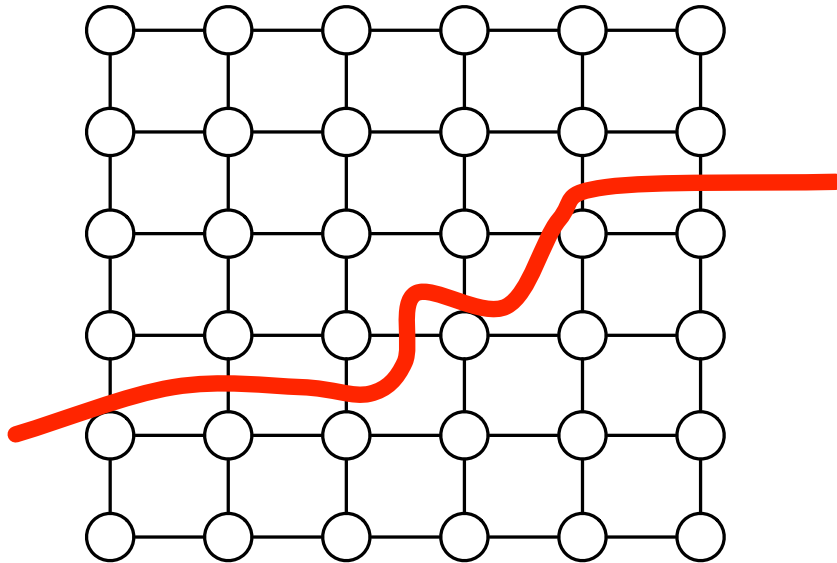
The second topology structure we explore in this thesis is a hexagon setup, as shown in Fig 3.3. This is a straightforward extension of the chained-form network, where multiple chains stacked together. This setup will give the target unlimited freedom in all direction

in a 2-D space.



**Figure 3.3:** Hexagon-shape network topology

Finally, we extend our tracking algorithm in a grid-shape structure as shown in Fig 3.4. This grid setup is an attempt to simplify the hexagon setup, where fewer connections are available for each sensor. However, this setup loses the fully connected triangular structure available in chained-form network and hexagon-shape network.



**Figure 3.4:** Grid-shape network topology

### 3.4 Two-tier Tracking Framework in Chained-form Networks

Chained-form network can be implemented along highways, bridges, waterways, or in the hallways of some buildings. In the chained-form network, each sensor are connected with four neighbors, while the target is in the chain, it may transit from one sensor to any of its neighboring sensors. The first tier of the algorithm will use this transition pattern and conditional probability to determine the rough target location. While given the target location from last time frame, the target location at this time frame will be estimated based on the transition of the target.

#### 3.4.1 Tier One - Viterbi Detection

Viterbi algorithm is used in the first tier of the algorithm to determine the rough location of the target. In this tier, a sensor that is closest to the target will be picked out, and is used as a cluster head to form a cluster. Viterbi algorithm is a distributed implementation of the Bayesian framework, where a state probability of the target is computed at each sensor location, and the sensor with the highest probability is selected.

Let take sensor  $i$  as an example. The target state probability for sensor  $i$  at time frame  $T$  can be computed as,

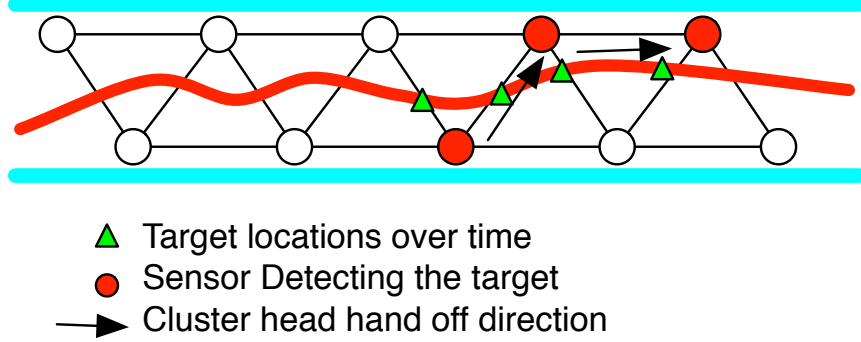
$$\delta_T(i) = \left[ \max_{j \in NB(i)} P(z_T^i | z_{T-1}^j) \delta_{T-1}(j) \right] P(y_T^i | z_T^i), \quad (3.4)$$

where,  $NB(i)$  indicates the neighbors of sensor  $i$ ,  $z$  is the true state of the target, and  $y$  is the observation made by the sensor. In this equation, three terms are involved. The first term  $P(z_T^i | z_{T-1}^j)$  is the transition probability, it can be derived as,

$$P(z_T^i | z_{T-1}^j) = \begin{cases} p & \text{if } j \in NB(i); \\ 0 & \text{if } j \notin NB(i). \end{cases} \quad (3.5)$$

It defines the probability that the target will travel from sensor  $j$  location to sensor  $i$  location at time frame  $T$ . In here, as shown by the equation, we restrict the transition of the target between neighboring sensors (i.e. in time frame, the target is only fast enough to reach the next hop sensor). The probability value  $p$  is a predefined constant, it can be the same for

all  $i, j$  pairs, or it can be altered based on empirical data. For generality, we assume it is the same over all sensor pairs, and for chained-form network, this value is set to  $1/5$  to be fair for all pair of neighbors, as shown in Fig 3.5. The second term in the target state



**Figure 3.5:** Target detection hand off between sensors in tier 1

probability equation is the target state probability of last time frame,  $T - 1$ . This term is self-explanatory, and it is inherited from previous iterations. The third term in the target state probability equation is the detection probability of the sensor, it can be defined as following

$$P(y^i|z^i) = \begin{cases} \eta & \text{if } y^i = 1; \\ 1 - \eta & \text{if } y^i = 0. \end{cases} \quad (3.6)$$

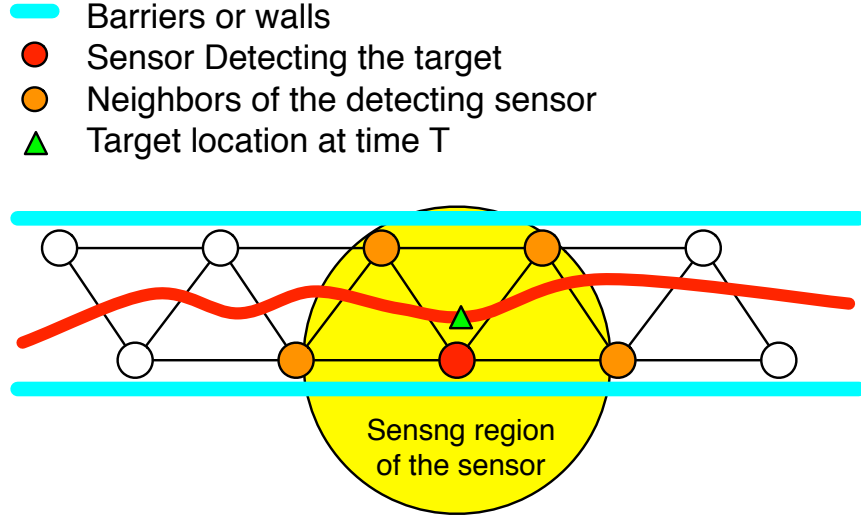
where,  $\eta$  is the probability of detection of sensor  $i$  given the target is near the sensor.

For offline processing, the  $\delta$  value for each sensor is computed and stored until the last time frame. The maximum  $\delta$  values for the last time frame are determined among all sensors, the sensor with that  $\delta$  value is considered the rough target location for the last time frame. Given this location, the algorithm then tracks backwards, find the maximum  $\delta$  for the previous time frame among the neighboring sensors of this location. For online processing, the  $\delta$  value is maximized at each time frame to obtain an on-time decision.

### 3.4.2 Tier Two - Maximum Likelihood Estimation

Once a sensor is selected in tier one, the algorithm is ready for tier two computation. A cluster is formed with the selected sensor and all of its neighbors. In the chained-form

network case, these clusters will include 5 sensors in a trapezoidal setup as shown in Fig 3.6. The selected sensor will be the cluster head while all the members of the cluster have 1-hop communication with the cluster head. All of the sensors in the cluster will then take the distance measure they obtained according to equation (3.3), and pass it to the cluster head. The distance measurement will be the actual distance value with some random noise. For this thesis, the random noise is assumed to be zero mean Gaussian.



**Figure 3.6:** Cluster formed around the detection result obtained from tier 1

The maximum likelihood estimation technique is used to estimate the precise location of the target. To estimate with the maximum likelihood technique, a likelihood function must be derived. In our algorithm, the likelihood function is derived by exploiting the independent noise distribution of the sensors in the cluster.

Equation (3.3) can then be rearrange to the following form,

$$n_i = m_i - r_i, \quad (3.7)$$

where  $i$  is the index of the sensor  $i$ . Since  $n$  is the zero mean Gaussian random noise, its distribution follows the Gaussian distribution,

$$P(n_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{n_i^2}{2\sigma^2} \right), \quad (3.8)$$

where  $\sigma^2$  is the variance of the noise. Substitute equation (3.7) for  $n_i$  in the Gaussian

distribution, we have

$$P(r_i|m_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{(m_i - r_i)^2}{2\sigma^2}, \quad (3.9)$$

This distribution can be computed for all of the sensors in the cluster, and since the noise are independent, these distributions can be multiplied together to found the joint distribution.

$$\begin{aligned} & P(r_i, r_{i+1}, \dots, r_{i+4} | m_i, m_{i+1}, \dots, m_{i+4}) \\ &= \left( \frac{1}{\sqrt{2\pi\sigma^2}} \right)^5 \exp \left[ - \sum_{k=i}^{i+4} \left( \frac{(m_k - r_k)^2}{2\sigma^2} \right) \right]. \end{aligned} \quad (3.10)$$

Since  $r_s$  are actual distance between the sensors and the target, it can be replaced using the coordinates of the sensors and the target, and hence we obtain our likelihood equation,

let

$$\begin{aligned} a(x_t, y_t) &= \\ & \sum_{k=i}^{i+4} \left( \frac{(m_k - (\sqrt{(x_k - x_t)^2 + (y_k - y_t)^2}))^2}{2\sigma^2} \right) \end{aligned} \quad (3.11)$$

then

$$\begin{aligned} L(x_t, y_t) &= P(x_t, y_t | m_i, m_{i+1}, \dots, m_{i+4}) \\ &= \left( \frac{1}{\sqrt{2\pi\sigma^2}} \right)^5 \exp[-a(x_t, y_t)]. \end{aligned} \quad (3.12)$$

The target coordinates  $(x_t, y_t)$  can be computed by maximize the likelihood function,

$$(x_t, y_t) = \arg \max_{x_t, y_t} L(x_t, y_t). \quad (3.13)$$

From the figures, we can easily see that we can easily construct the likelihood function with readings only from 3 of the sensors, instead of using all 5 sensors in the neighborhood. However, if only 3 sensors are used, if one of the sensor has an outlying measurement



reading, the tracking result will be affected greatly. 5 sensors will effectively mitigate this problem, using more sensors will be a little redundant with limited improvement, but huge communication burden, since multi-hop communication will be needed.

## **3.5 Two-tier Tracking in 2-Dimensional Network Topology**

### **3.5.1 Hexagon-shape Network Topology**

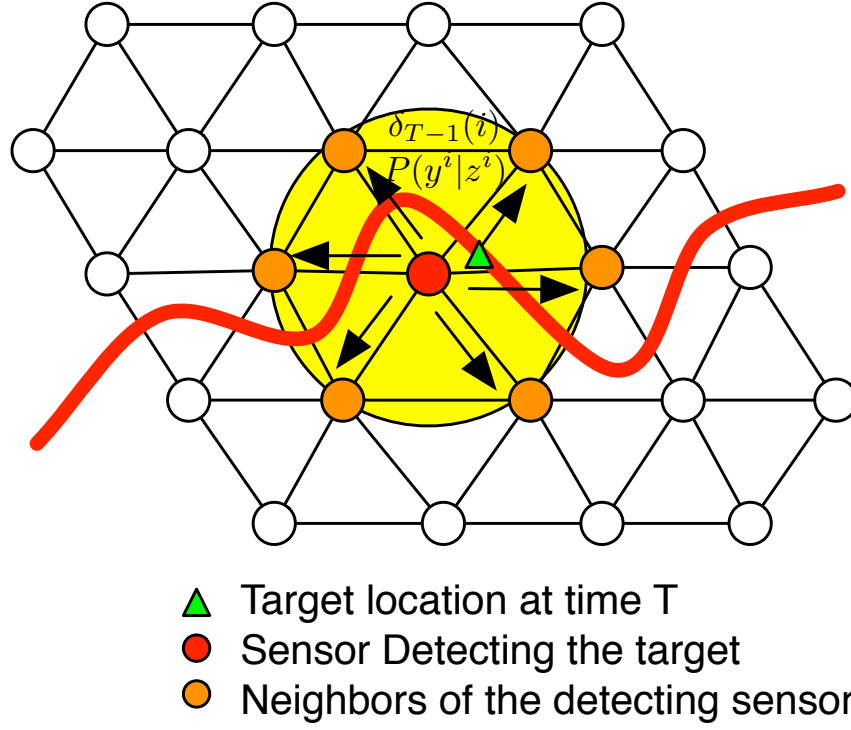
Chained-form network can only cover a narrow passive, for tracking in an open space, the chained-form network need to be extended to a full 2D space topology. The most intuitive extension of the chained-form network is a hexagon-shape network. As shown in Fig 3.7, the hexagon-shape network preserved most features of the chained-form network, it is several chained-form network stacked together. To track in this hexagon-shape network is not very different under the chained-form network framework. The only difference is that the number of neighboring sensors changed from 4 to 6.

The Viterbi algorithm in the first tier stays almost exactly the same, except that in this case, target would have more moving options since there are more neighbors connected to each sensor. In this case, the target may transit to 6 directions, with the probability of  $1/7$  in each direction, and the remaining  $1/7$  is the probability that the target will remain near the same sensor in the next time frame.

Once the sensor is selected in tier 1, the cluster will be formed in tier 2. In the hexagon-shape network, the cluster will include the cluster head with all six of its neighbors, which is a hexagon shape in the topology. The likelihood function can be computed using all of the sensors readings in the hexagon shape, and the likelihood function is maximized to find the target location.

### **3.5.2 Grid-shape Network Topology**

Although hexagon-shape network is able to extend the chained-form network to a 2D space, however, it is a little redundant to use readings from 7 sensors to perform the maximum likelihood estimation. Deploy sensors in to a perfect hexagon-shape autonomously is

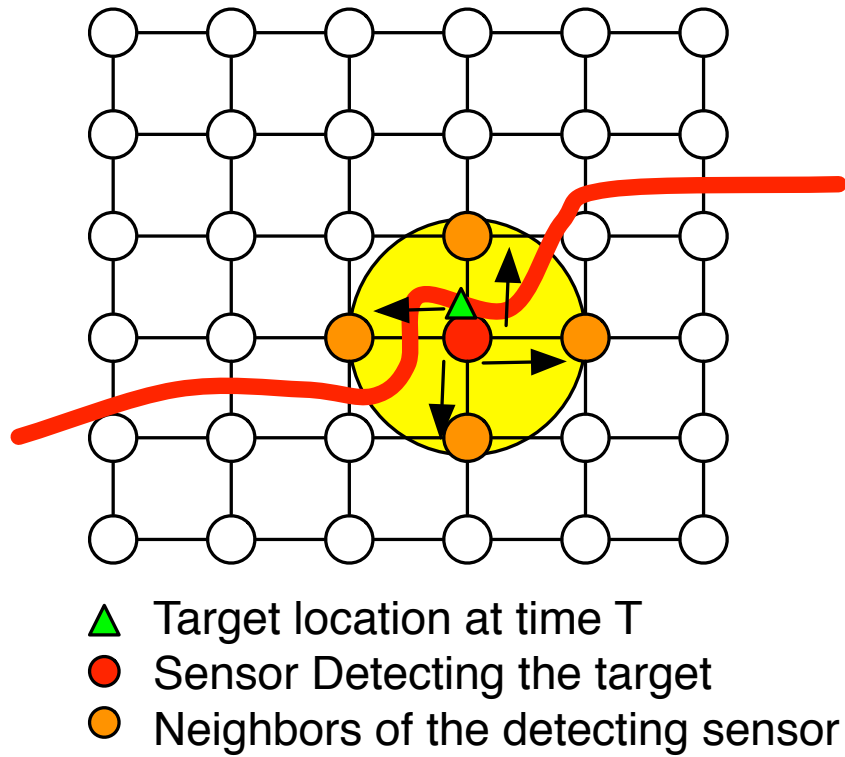


**Figure 3.7:** Message passing in hexagon-shape network with sensing region defined

tedious. Grid-shape network can be another option to perform tracking in 2-D space. As show in Fig 3.8, in the grid-shape network, there is only 4 neighbors for each sensor, a similar neighborhood setup as the chained-form network, hence the computation cost will be very similar to the chained-form network case.

Tier 1 of the algorithm for grid-shape network will be identical to the chained-form network, the sensor detection range threshold will be set similar to Fig. 3.8, note that under this sensing region assumption, there are only 4 neighbors involved. Although the algorithm is identical to the chained-form network, the detection result obtained in this network topology will be less accurate. The situation illustrate in Fig. 3.9 will result in detection inaccuracy. As shown in the figure, from time  $T1$  to  $T2$ , the target transit from sensor  $i$  directly to sensor  $k$ , however, by assume sensor can only transit between neighbors and the definition of neighborhood in grid-shape network, the detection result for time  $T2$  is most likely to be sensor  $j$  instead of sensor  $k$ .

Although inaccurate result may be obtained in tier 1, however, the inaccuracy is always bearable when we construct the likelihood function, since the “real” solution for tier 1 will always be included in the cluster in tier 2, only that it will not be the cluster head. The



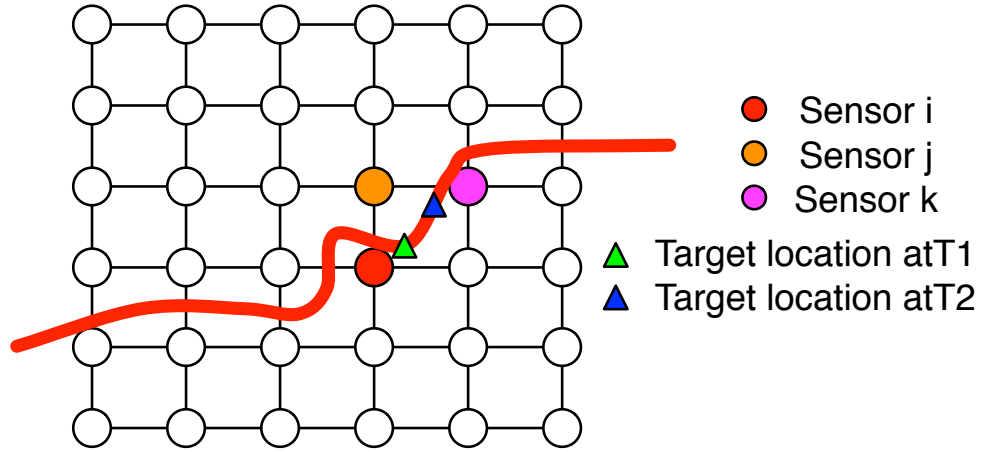
**Figure 3.8:** Message passing in grid-shape network with sensing region defined

result in tier 2 will not be altered, even this small inaccuracy happened in tier 1.

## 3.6 Simulation

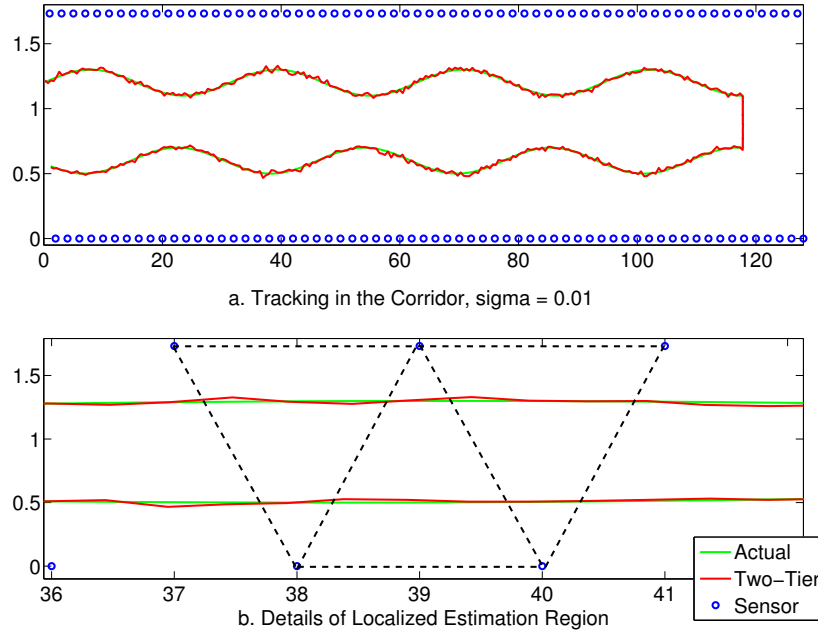
### 3.6.1 Chained-form Network

The first simulation is a general tracking setup in a corridor. A 128 meters long and 1.8 meters wide corridor is considered. Sensors are laid out along the boundary of the corridor, forming equal lateral triangles. Each node is 2 meters apart from any of its neighbor. The sensors are assumed to have disk shaped detection region with radius of 2 meters. Target traverse through the field from left most of the region to the right most near the upper bound of the corridor, and it then turn back return to the left most near the lower bound of the corridor. A small amplitude sine wave is added to the target's trice to increase path



**Figure 3.9:** Detection inaccuracy due to grid-shape network

variation. The 5 node configuration is used at the Maximum Likelihood phase. Multiple  $\sigma$  values for noise is applied to the system. The tracking result is shown in the following figure.

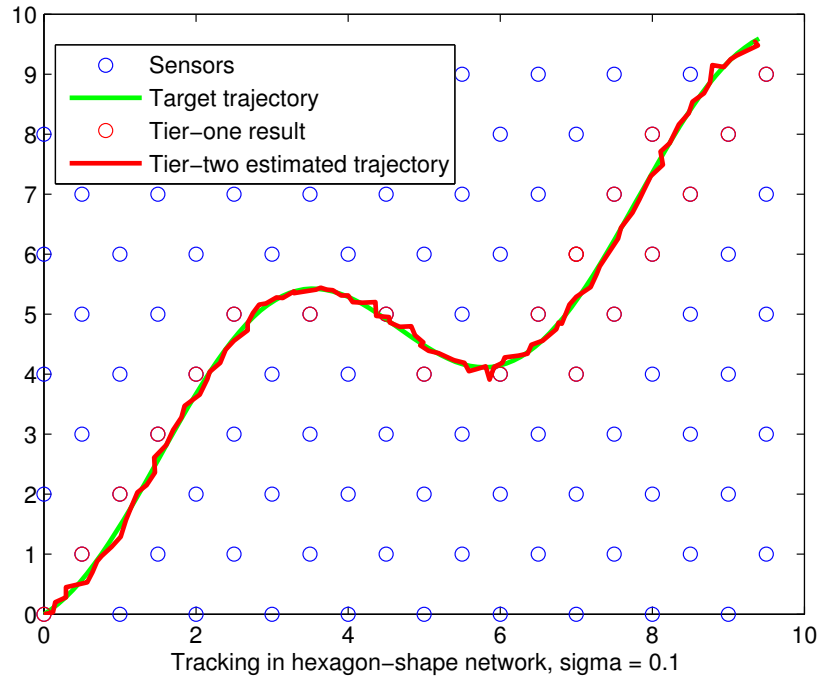


**Figure 3.10:** Tracking in a chained-form network,  $\sigma^2 = 0.01$

Subfigure (b) is a zoomed in version to demonstrate the sensors involved in a typical maximum likelihood estimation phase in tier-two.

### 3.6.2 Hexagon-shape Network

The hexagon case is simulated, the hexagon case is a direct extension to the corridor case into a full 2-D space. In this simulation, 100 sensors are deployed in a space of size 10 by 10 meters. The average distance between any two sensors are 1.2 meters. The sensors are assumed to have disk shaped detection region with radius of 1 meter. The target traverse through the field from the lower left corner to the upper right corner following a sinusoidal path. In the second tier, the sensor from tier-one and all six of its neighbor are involved in the maximum likelihood computation. The tracking result for the hexagon-shape network is shown in the following figure.

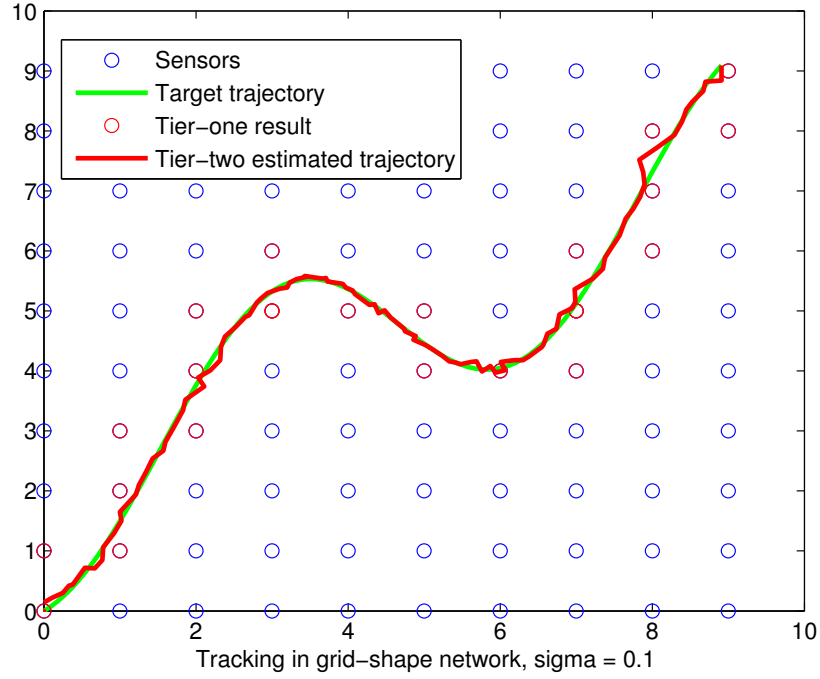


**Figure 3.11:** Tracking in a hexagon-shape network,  $\sigma^2 = 0.01$

### 3.6.3 Grid-shape Network

Finally, a grid-shaped network is simulated. The grid-shaped network is somewhat unique since given the center node, none of its neighbors are interconnected, but this has little impact on the maximum likelihood phase in tier-two, since we are manipulate the probability density of the measurement noise, and the measurement noise are independent

to each other either way. 100 sensors are again deployed in a 10 by 10 meter space. The distances between each pair of sensors are 1 meters. This number is again used as the detection range of the sensor for tier-one. The target follows the same trajectory as shown in the hexagon-shape case. The tracking results are shown in the following figure.



**Figure 3.12:** Tracking in a grid-shape network,  $\sigma^2 = 0.01$

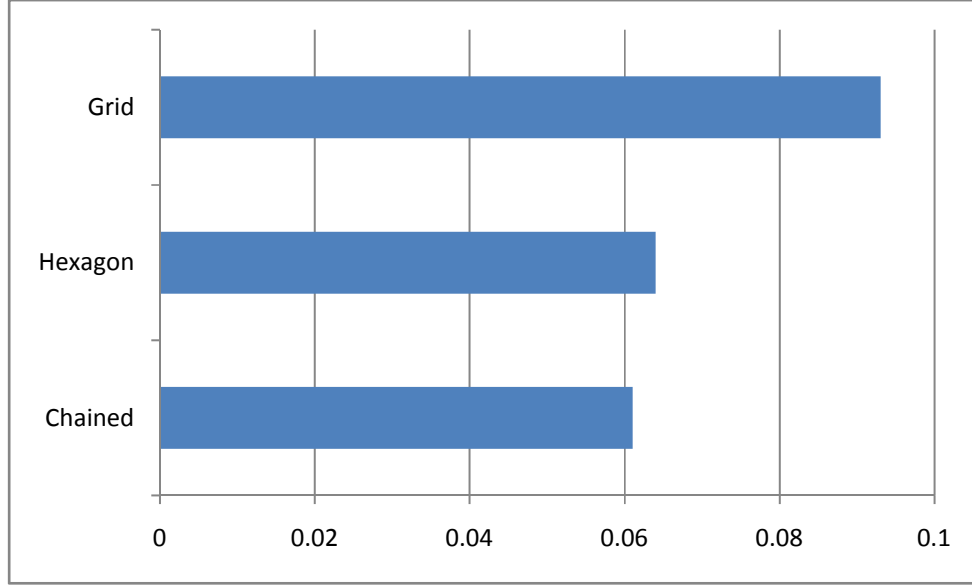
### 3.6.4 Tracking Errors

Tracking error for each of the case is computed according to

$$E_{est} = \frac{\sqrt{\sum_{t=1}^N [(x_{est}^t - x_{act})^2 + (y_{est}^t - y_{act})^2]}}{N} \quad (3.14)$$

where  $(x_{est}^t, y_{est}^t)$  is the estimation, and  $(x_{act}, y_{act})$  is the actual coordinates of the target. The tracking error for each case is show in the following figure.

As we can see, tracking in chained-form network and tracking in hexagon-shape network yields similar error rates, since the interconnection scheme of the sensors are essentially the same for both cases. The error for the grid-shape network case is much more significant



**Figure 3.13:** Tracking error for Chained-form network, hexagon-shape network and grid-shape network

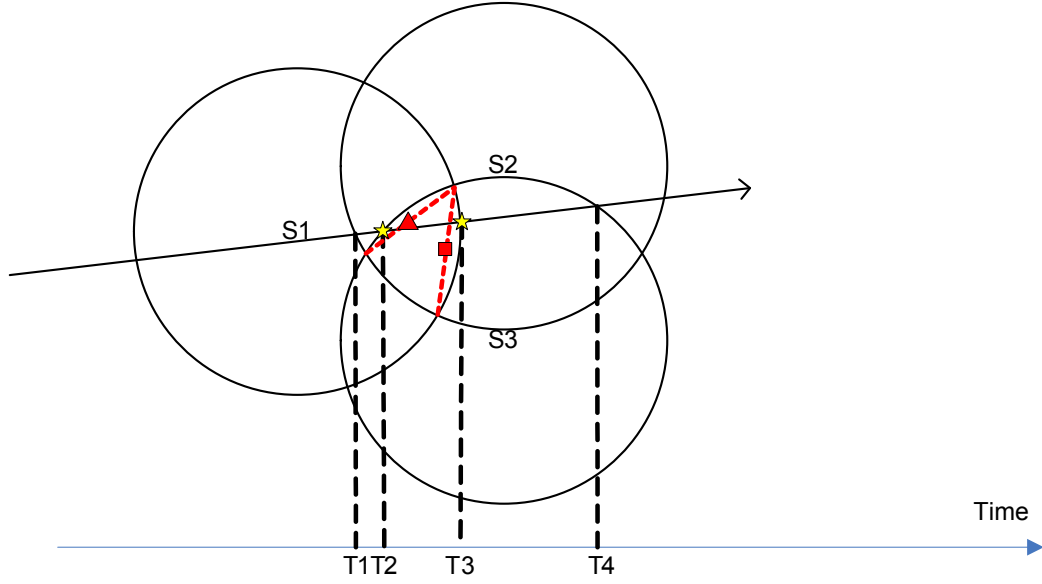
because of the problem described in Fig. 3.9.

### 3.6.5 Comparison to Similar Tracking Schemes

In “Compact distributed target-tracking algorithm with binary sensors” [58], the authors proposed a similar lightweight target tracking algorithm using binary sensor network. However, instead of using a two-tiered method, they used only binary sensors. This will degrade the localization accuracy, however, the authors took an extra step after forming the cluster to improve the accuracy. A “triplet circle intersection” (TCI) method is proposed in that paper.

As shown in figure 3.14 , the TCI method monitors when the target enters the detection region of each sensor in the clusters. T1 represent when target entering the sensing region of S1 and S2, T2 represent the target entering the sensing region of all three sensors, T3 represent the target exiting the sensing region of S1, and T4 represent the target exiting the sensing region of the S3. In TCI, only T2 and T3 are interested, since they are covered by all three sensor, hence the “triplet circle”. At time T2, the star represent the actual location of the target, and TCI compute the intersection arc, draw the chord along that arc, and use the mid-point of the chord as the estimated target location, represented by the triangle. Similar for T3, where the square is the estimated target location, while the star is the actual

location.



**Figure 3.14:** Triplet circle intersection (TCI) method

When we compare the proposed two-tier framework to the TCI method with the binary sensor scheme, we can find the two-tier method achieves better estimation accuracy with similar computation cost, and less constraints. While direct comparing numbers to numbers does not make sense, the accuracy achieved is in centimeter range while the TCI method is in meter range. We would like to examine the possible source of error for each method.

The only source of error in the two-tier method is the noise in the sensor measurement. In our simulation, we assumed Gaussian noise with  $\sigma = 0.1$  and  $\sigma = 0.5$ , and even with a high  $\sigma$ , the mean square error is still in the centimeter range. In TCI method, first of all, the sensor noise is not considered since they assume binary sensor with perfect fixed sensing range. The second source of error is that depends where the target is while entering/exiting the triplet region, the error can range from very small to half of the chord length. This introduces more uncertainty to the result, and makes the accuracy heavily dependent on the target trajectory, which is completely uncontrollable. The other source of error is that if the sensor network is not uniformly distributed, as two sensors get closer, their intersecting arc also becomes longer, hence the mid-point of the arc will be representing a larger triplet circle area, hence the accuracy actually decreases with densely distributed sensor network. The two-tier method does not suffer from this at all.

Computation cost wise, for the TCI method, the sensor need to solve a set of simultaneous equation to find the intersection points on the circle. While for the two-tier method, there is only one inverse likelihood function to minimize. Hence the computation cost



is comparable. The two-tier method result accuracy can be improved by including more sensors readings in the likelihood function, and the additional cost is very small. The TCI method does not have this kind of scalability.

### **3.7 Conclusion**

A new two-tier low-complexity target tracking framework for distributed sensor networks is proposed in this thesis. The framework is fast, low cost, and can be applied to different sensor topology under different space scenarios, it can be applied to both regularly and irregularly distributed network topology. Simulations are performed to demonstrate the algorithm. The framework shows that as the network expand, the computation cost only increase marginally. The framework can be implemented on an ultrasonic sensor network platform to track mobile targets traveling in the network.

# Chapter 4

## Target Tracking in Markov Random Field <sup>1</sup>

### 4.1 Introduction

Current sensor technologies enable us to fabricate small and inexpensive sensor-rich devices that can be used in various intelligent surveillance applications. With limited communication capability, the sensors can be linked together into a mesh network, hence provide a convenient test-bed for in network signal processing and distributed parallel algorithms. A classic application of the sensor network is target tracking. It can be found in many real life applications such as missile defense, battlefield situation awareness, air traffic control, building surveillance, emergency response and many emerging applications such as supply chain management and wild inhabitant monitoring.

Many tracking algorithms have been developed. These algorithms can be generally classified into three categories: probabilistic data-target association, binary sensor detection and random sampling. Probabilistic data-target association is the process of mapping observations to the target through probabilistic analysis of the system [14], [59], [60]. Full sensor data is usually required for accurate analysis, which will cause heavy computation and communication burden to the system. The binary sensor detection approach sacrifice accuracy for simplicity. It treats every sensor reading as a binary detection rather than a

---

<sup>1</sup>©2009 IEEE. Portions reprinted with permission, from **Lufeng Shi**, Jindong Tan, and Zhijun Zhao, “*Target tracking in sensor networks using statistical graphical models*” , in Proceedings of IEEE International Conference on Robotics and Biomimetics, pp. 2050 - 2055, 2009. See Appendix for a copy of the copyright permission from IEEE.

ranging estimation, and follows the target by clustering and analyzing the target transition probabilistically. There are many cluster based algorithms developed with binary sensors [26], [61], [62]. Random sampling can be considered an extension to the probabilistic data-target association. Its essence is still measurement to target association. The typical ones includes Particle filtering and Markov Chain Monte Carlo Data Association (MCMCDA) [20], [19], [63], [64]. These approaches use random samples to represent probabilistic distributions when analytical solutions cannot be found for the classic estimation approach.

Most of these existing algorithms are reincarnation of known centralized algorithms in distributed fashion. Although they may be implemented in a distributed fashion, network topology, scalability and data fusion are rarely considered. It can be easily shown that the complexity of some algorithms increase exponentially as the size of the network grows. At the same time, most of these algorithms are not truly distributed. They do not aggregate data while propagating, but rather, the data is transferred to a central processor or a cluster head to be processed. In many cases, the direct links between data collectors and cluster heads are assumed. This assumption reduces the need for information forwarding management, however, it is not realistic in real networks with complex topologies.

In this thesis, a graph based tracking algorithm is developed. The algorithm effectively associates the topology of the sensor network to a statistical graphical model. This graphical model not only well captures the characteristics of the network topology, but also enables the use of mature graph theory techniques to perform data fusion and make inferences in the sensor network. In our algorithm, the network is triangulated into three nodes cliques, thus reduces the complex, loopy random distributed sensor network into a simple, loop free clique tree. Belief propagation is applied to this clique tree to compute a joint probability distribution, the target location is discovered by maximize the distribution. Belief propagation is a two-way message passing algorithm, which not only manages the information propagation on the clique tree, but also enable each clique to perform partial calculates in order to obtain the final result.

## 4.2 Related Works

Many of the existing tracking algorithms are reviewed in this section for comparison. Classic tracking algorithms including Multiple Hypotheses Tracking (MHT), Joint Probabilistic Data Association (JPDA), binary sensor detection, particle methods and graph based methods are studied.

Multiple Hypotheses Tracking (MHT) algorithm is one of the earlier methods for solving

data association problems. This algorithm exhaustively enumerates out all possible locations of the target, and computes the associated probability to each of these hypotheses. Decision will be made in the end based on probability distribution of the hypotheses. During the process, all hypotheses are kept for delayed decision, which means the number of hypothesis increase exponentially over time, hence will cause heavy computation and communication burden. The delayed decision is also not suitable for real time processing. Detailed MHT algorithm description can be find in [14], [65]. This approach is usually infeasible to implement due to the exponential growth of the number of hypotheses.

Joint Probabilistic Data Association (JPDA) described in [59] is a variation of the MHT algorithm. It still enumerates out all possibilities of the target, however, the decision is made at each time period and other hypotheses are discarded. This will effectively control the growth of the number of hypotheses. A detailed tutorial on JPDA in multiple target tracking is given in [60], where the hypotheses collapse into a single posterior distribution function at each time frame. The tracking problem is formulated into an iterative, two step Bayesian filtering process. The first step predicts the future state of the target with known target dynamics, the second step updates the predicted state with the new measurements. The limitation of JPDA is that the number of targets has to be fixed and known in advance. There are some variations to the JPDA algorithm available, which allow number of targets change during the process [66]. However, the change rate is strictly limited to at most one target per time frame.

Binary sensors are used to reduce the computation complexity, and communication load. Each sensor only needs to report single bit information about on the presence of the target. However, since only the binary readings are used, the result can only reflect the location of the nearest sensor to the target. Viterbi algorithm is used in [62] is a typical application of binary sensor in target tracking. It assumes binary sensors with disjoint sensing regions. It calculates a discrete probability mass distribution functions over all sensors, and select the sensor location with the maximum probability as the tracking result. In this case, the sensor reading is used as a weight, i.e. detection indicates higher weight on the sensor while non-detection means lower weight, rather than computing the range of the target for non-binary sensors. While still assuming binary sensors, researchers make use of the topology of the sensor network to refine the target location through clustering [26], [17]. For example, In [17], the author finds the union of the sensing regions of all the detecting sensors, and subtract it from the union of sensing region of all the non-detecting sensors, and claim the target is located inside the resulted region. Other cluster based algorithm such as Voronoi diagram described in [61], and weight based system described in [67] were also developed. This does improves the tracking accuracy, but the improvement is minor.

In ideal cases, where unknowns can be modeled by parametric distributions such as Gaussian or Poisson, MHT and JPDA will always provide good tracking result. In reality, however, analytical solution is not always feasible. Sample based algorithms are developed

to provide numerical solutions. Particle filtering algorithm, also known as sequential Monte Carlo algorithm, is a typical example of sample based algorithms. It can be considered as an extension of JPDA approaches. Particle filter conducts regular two step Bayesian filtering process with non-parametric distribution functions. The distribution function is represented by a number of sample points with associated weights. In cases where analytical solution cannot be found, this approach produces good approximation with relatively low cost [20], [19].

MCMCDA is another data association approach using particle method. In MCMCDA, a stationary Markov chain is assumed which ensures the temporal independence and the time invariant. Association hypothesis are still made, however, the hypothesis are selected at random, and evaluated by comparison with the random samples generated from a proposal function [22], [15]. According to the result, hypotheses may be accepted, discarded or modified. The algorithm loops through these hypotheses making, selecting and altering over all measurements and decide an acceptable association pattern. In terms of tracking these steps are reflected as the birth, death, split and union of target tracks, examples can be found in [63], [64]. The MCMCDA uses random sampling strategy to evaluate hypotheses while in MHT, all hypotheses are exhaustively computed and evaluated. Thus, MCMCDA has indeed improved the MHT approaches in the sense of avoiding hypotheses explosions, but this algorithm loses the optimality of the MHT. Since it still makes delayed decision, it is improper for obtaining timely information about the target, thus not suitable for a real time tracking system.

Graphical model and message passing algorithms has been used in visual tracking applications [68], [69]. A general framework in solving Bayesian inference problem for visual tracking under non-linear, non-Gaussian situations is developed in [68]. It provides a numerical solution to an analytically infeasible problem. It is an image processing technique rather than following an object in a physical sensor network. The Dynamic Bayesian Network (DBN) model is constructed using the transition of the states in the image rather than capture the topology of a sensor network. The inferences are made using the standard two step prediction-update Bayesian filter. In [69], although a few cameras are used to collect data, but they are used just to provided a broader view, rather than forming a physical network, hence the graphical model is still not a reflection of the physical network topology. Since this paper focuses on tracking the position of football players on a football field, the vertices on graphical model are the possible positions of the players and links are the possibility of transition between these positions. Both of the algorithms are addressing image processing techniques rather than data fusion in a physical sensor network. In this thesis, the graphical model is used to capture the topology of a physical sensor network, and the inference algorithm is used to process data while propagation.

### 4.3 Problem Formulation

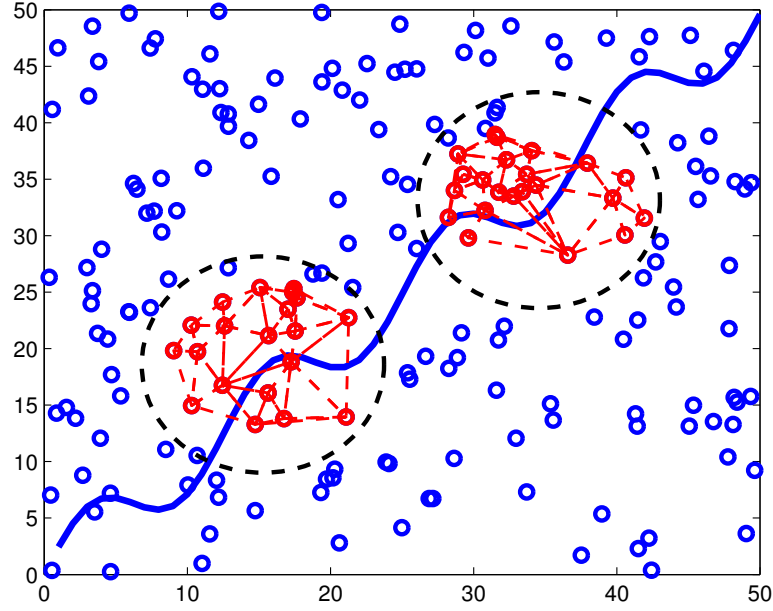
To effectively track a target in sensor network, the network needs to be localized and synchronized. The sensors should be able to measure the rough distance between the sensor and the target. The measurement model can be formulated as a Gaussian distribution centered at the measured location. At each time frame, with multiple sensors detecting the same target, we compute a joint probability distribution of these sensors using a joint Gaussian distribution model:

$$p(x) \propto \exp\left(-\frac{1}{2}(x-m)^T C^{-1}(x-m)\right) \quad (4.1)$$

where,  $m$  is the vector of different sensor measurement, and  $C$  is the covariance matrix. The location of the target at this time frame can be inferred by maximizing this joint distribution.

Making inference on a random distributed network is very complicated. Fig. 4.1 demonstrates such challenge, the sensors in the two circles may be observing two different targets, or they may be looking at the same target just at two different times. How to associate the data generated in the two circles is always challenging. When the two circles get close enough so that they overlap, the relations between the sensor measurements become more ambiguous. With a moving target, the group of sensor that detects the target is dynamic, in randomly distributed topologies, this change cannot be predicted. In summary, there are two major challenges needed to be addressed to conduct high accuracy target tracking in sensor networks using graphical model. First challenge is to organize the randomly distributed network topology into a manageable structure. The second challenge is to be able to make inference while passing the data.

Belief propagation on a clique tree is a good solution to both of the challenges. Clique tree algorithm organizes the randomly distributed sensors into a tree structure, while belief propagation fuses data into a joint distribution while passing information on this tree structure.



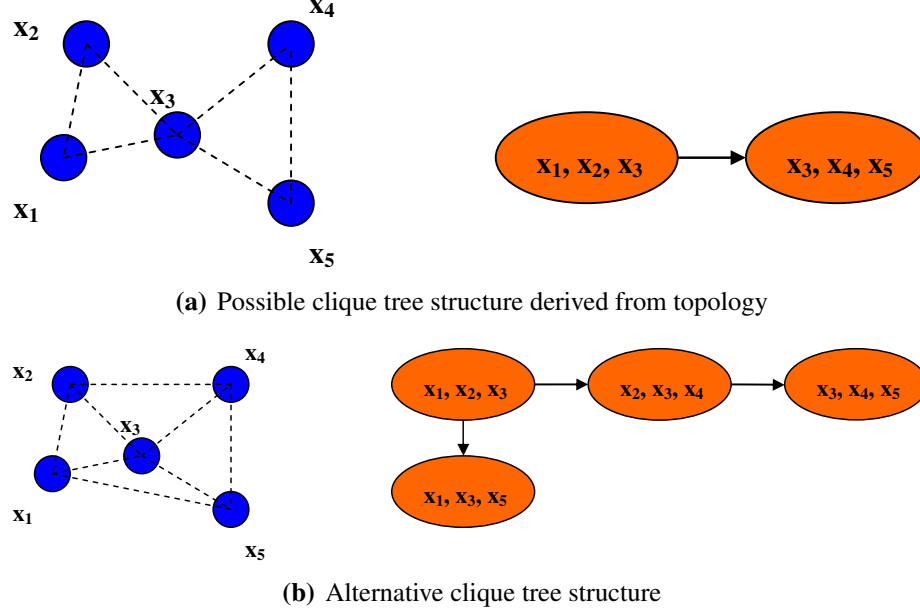
**Figure 4.1:** Challenges in random distributed sensor networks

## 4.4 Algorithm Formulation

On a clique tree, the interconnected sensors are represented with a single “super” node called cliques, and the cliques are linked together into a tree structure. This not only simplifies the graph topology by representing the original one with fewer nodes, but also eliminates loops in the topology. It is essential for message passing algorithms to make exact inference and converge in finite times. Fig. 4.2 is an example of the construction of a clique tree. The figure showed 3 possible ways to construct the clique tree, and they are equally valid. In practice, however, the last configuration is often used. If the first two configurations are used, then for a target to transit directly between two cliques, it will have to pass exactly through the node in the middle, which is statistically impossible.

Each clique is modeled by the joint probability distribution of the nodes the clique contains. This joint probability is usually called clique potential. It can be computed according to Hammersley-Clifford theorem:

$$p(x) = \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j). \quad (4.2)$$



**Figure 4.2:** Example of network topology and associated clique trees

where,  $\psi_i(x_i)$  is the node potential,  $\psi_{ij}(x_i, x_j)$  is the link potential,  $V$  is the set of vertices, and  $E$  are the set of edges.

With clique tree structure set up, message passing algorithm can be applied to make exact inference in the network. The message initiate at the leaf node using the leaf node potential. The message is passed up to its parent. The parent convolute the message received with its local potential and forward link potential to generate a new message, and this new message is passed up to the upper layer. A marginal probability distribution can also be computed upon receiving the message. The equations are:

$$m_{i \rightarrow j}(x_j) = \int \psi_{ij}(x_i, x_j) \psi_i(x_i) m_{k \rightarrow i}(x_i) dx_i \quad (4.3)$$

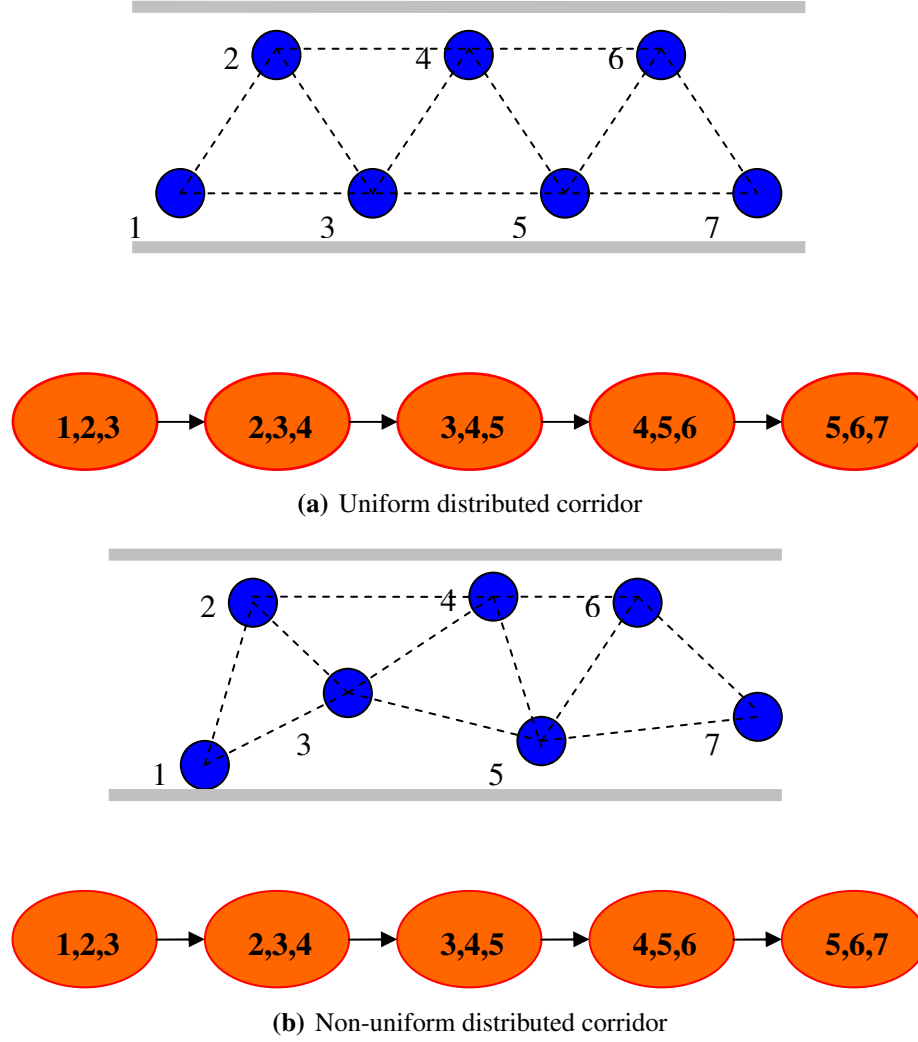
and

$$p_i(x_i) \propto \psi_i(x_i) m_{k \rightarrow i}(x_i) \quad (4.4)$$

where equation 4.3 calculates the message to be passed to the parent, and equation 4.4 calculate the local marginal.  $k, i, j$  are node indices, and their relation is that  $k$  is the child of  $i$  and  $i$  is the child of  $j$ , thus the message is passed from  $k$  to  $i$ , then from  $i$  to  $j$ .

In this thesis, we are particularly interested at tracking in two types of network topology.





**Figure 4.3:** Clique trees for uniform and non-uniform chained-form regions

First type is a chained-form network, the sensors are distributed along a narrow corridor, as shown in Fig. 4.3. This type of topology suits many of the real world situations including tunnels, bridges, canals, and many other narrow passages. The second structure we considered is a 2-D randomly distributed sensor field, as shown in Fig. 4.1. This structure best suits the applications in an open area, such as rooms, city square, or battlefield.

#### 4.4.1 Chained-form network

chained-form network can be considered a special case of 2-D networks, where as it has unlimited freedom in forward and backward directions, and the movement in other two

directions are restrained. In chained-form networks, the triangulation comes naturally as shown in fig. 4.3(a). In fact, even if the sensors are not laid out uniformly, the chain remains, and the clique can be constructed exactly the same way. Fig. 4.3(b) demonstrates such a situation.

#### 4.4.1.1 Node and link potential

In order to do belief propagation, we have to define the node potential and link potential. Since we have assume the sensors have a joint Gaussian distribution, the single node potential can be formulated as a regular Gaussian distribution

$$\psi_i(x_i) \propto \exp\left(-\frac{1}{2}C_{ii}^{-1}(x_i - m_i)^2\right). \quad (4.5)$$

The link potential can be formulated in a similar fashion in terms of the covariance matrix

$$\psi_{ij}(x_i, x_j) = \exp(-x_i C_{ij} x_j) \quad (4.6)$$

#### 4.4.1.2 Message passing

The framework of message passing is described in equation 4.3 and 4.4. In our particular case, since we have a clique tree structure, each node on the tree represents a clique, rather than a single sensor. The clique potential can be obtained using the Hammersley-Clifford theorem:

$$\psi_{Cl}(x_{Cl}) = \frac{1}{Z} \prod_{i \in Cl} \psi_i(x_i) \prod_{ij \in E} \psi_{ij}(x_i, x_j) \quad (4.7)$$

where the node potential and link potential are defined in equation 4.5 and 4.6.

Link potential for clique tree is another important factor in message passing algorithms. However, the link potential between cliques is somewhat different from the link potential between two sensor nodes. In the physical network, the link potential between sensors is highly dependent on the physical distance between them. In the clique tree constructed, the link between cliques does not contain any distance information, but rather represent transition between cliques. In this thesis, we assume the link potential between cliques to be unity, meaning the target is equal likely to transit into the vicinity any of the neighboring cliques.

#### 4.4.1.3 Inferences on clique tree

Once the joint distribution is obtained with the message passing algorithm, the target location can be inferred. The sensors are assumed to measure distance, and the distance can be represented by an  $(x,y)$  coordinate by substitute in  $d = \sqrt{(x-x_i)^2 + (y-y_i)^2}$ , where  $d$  is the distance,  $(x,y)$  are the coordinate of the target and  $(x_i, y_i)$  are the coordinate of the sensor (i). A  $\arg \max$  operator can be used to find the best target location in this time frame.

In centralized system, this maximization process can be carried out easily using Newton's method or the steepest decent method. However, for a distributed system, the central processor that contains all of the information does not exist. Luckily, with a small modification of the message passing framework, a distributed way of maximization can be found. At each leaf clique, instead of passing the entire clique potential as a distribution function, the clique potential is locally maximized, and a so called maximum message is generated and passed to the upper layer.

$$m_{i \rightarrow j}(x_j) = \max_{x_i} (\psi_{ij}(x_i, x_j) \psi_i(x_i) m_{k \rightarrow j}(x_i)). \quad (4.8)$$

Upon receiving this maximum message, each of the cliques will calculate their local maximum, and use it to generate the out-going messages

$$x_i^* = \arg \max_{x_i} \left( \psi_i(x_i) \prod_{k \in NB(i)} m_{k \rightarrow i}(x_i) \right). \quad (4.9)$$

Once the maximum message reaches the root of the clique tree, each clique should have a local maximum value ready, and this value is passed back from the root to the leaves. Comparison is done on the way back to find the global maximum value. This is also known as the max-product algorithm on trees [70].

#### 4.4.2 Random distributed network

In random distributed networks, the formulation of the clique potentials and the framework for the message passing algorithms are exactly the same as the chained-form network. The only difference is to that the clique tree structure is not obvious as the chained-form network. The challenge in the random distributed network is to build this clique tree structure.

In this thesis, we take a dynamic approach to build up the chain shape in the sensor network.

Assume the initial location of the target and the Delaunay triangles of the network are known. At next time frame, the three neighboring triangles that share edges with this starting triangle are included into the group, thus we have a small clique tree with 4 cliques. Inference is made on this small tree, and the target is localized in a particular triangle, and in the neighbors that share edges with this triangle will be included in the clique tree in future time frames. In here, neighbors of a triangle imply the three triangles that are sharing lateral with it. The algorithm is shown in algorithm 1.

---

**Algorithm 1** Algorithm for constructing clique tree in random network

---

```

j = Current_Target_Location
Chain_Region = [j]
for time = 1 to Tmax do
  for i ∈ DelaunayTriangles do
    if i is NB(j) then
      Chain_Region = [Chain_Region, i]
    end if
  end for
  Apply message passing inference on Chain_Region
  j = New_Target_Location
end for

```

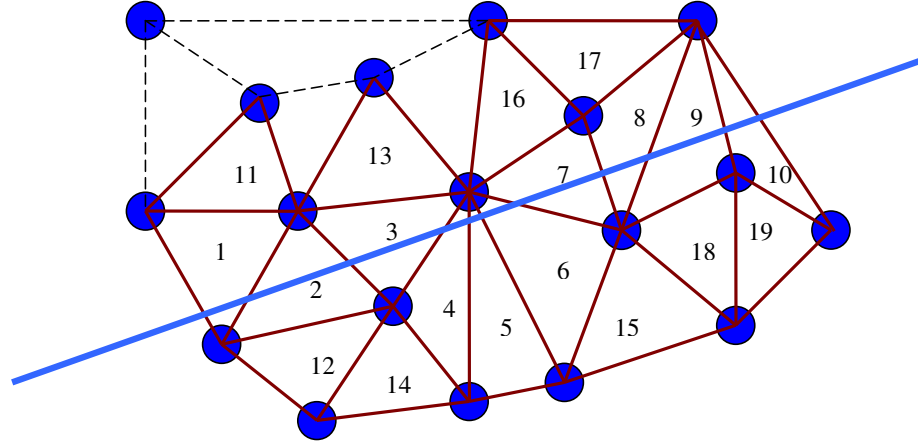
---

The size of the clique tree grows as the time increases, at each time frame, at most two new cliques will be included into the clique tree. Fig. 4.4 demonstrate a constructed clique tree structure in a random distributed network. This clique tree building process is automated in the simulation.

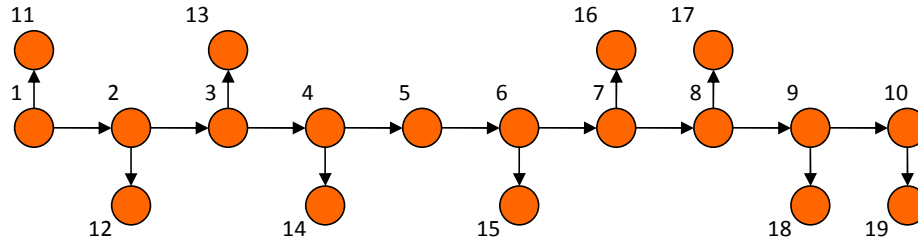
Note that the clique tree shown in fig. 4.4 is similar to the chained-form network shown in previous sections but not entirely the same. In the chained-form network, the clique tree appears to be a one-dimensional line, while in this case the tree has branches. With these branches, a node will have to gather the message from all its children to construct its own message to its parent. The message formulation for a typical node with two branches will be changed into the following form

$$m_{i \rightarrow j}(x_j) = \max_{x_i} \left( \psi_{ij}(x_i, x_j) \psi_i(x_i) \prod_{k \in NB(i) \setminus j} m_{k \rightarrow j}(x_i) \right) \quad (4.10)$$

where *j* is the parent of *i*, and *k* are the neighbors of *i* exclude *j*. The message node *i* pass to node *j* includes the product of the messages from all of *i*'s children.



(a) Topology of a random distributed network



(b) Junction tree associated with the topology

**Figure 4.4:** Corridor obtained in a random distributed network

## 4.5 Simulation

We evaluated the performance of this graph based tracking algorithm by simulations. Three situations were considered, tracking in the corridor case with uniformly distributed sensors, in the corridor with non-uniform distributed sensors and in a random distributed sensor networks. In addition to construct the target trajectory, the estimation error was computed according to

$$E_{est} = \frac{\sum_{t=1}^N \sqrt{[(x(t) - x_{act}(t))^2 + (y(t) - y_{act}(t))^2]}}{N}, \quad (4.11)$$

where  $(x(t), y(t))$  is the estimation result at time  $t$ , and  $(x_{act}(t), y_{act}(t))$  is the actual coordinates of the target.

### 4.5.1 Uniform Corridor

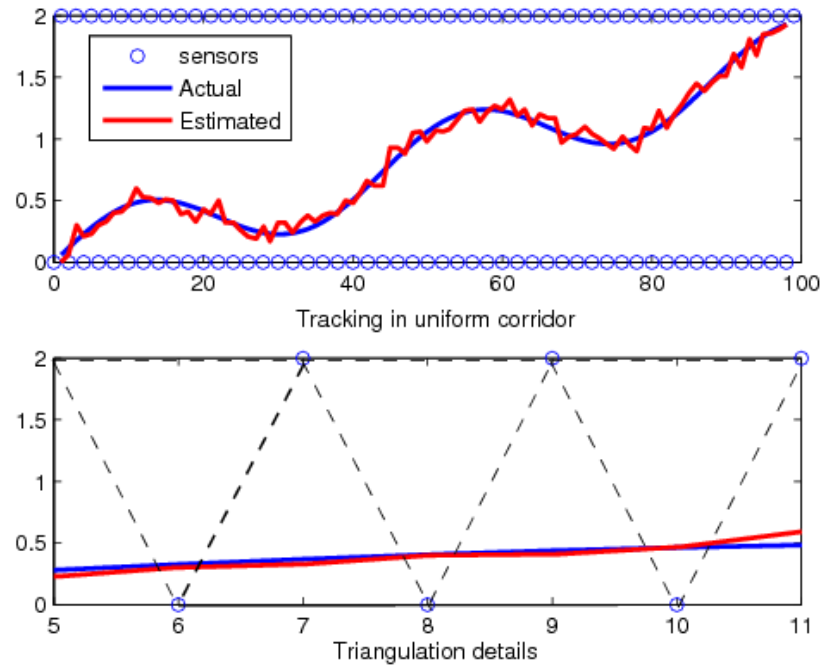
In this simulation, 100 sensors were evenly distributed along the two boundaries of a 2-meter by 100-meter rectangular region. A zero mean Gaussian measurement noise with variance of 0.1 was applied to the measures. The target follows a sinusoid path originated from the lower left corner of the corridor to the upper right corner. The tracking result is shown in fig. 4.5(a). The circles along the two boundaries are sensor nodes, the thick solid line is the actual trajectory of the target, and the dashed line is the estimated trajectory. The estimation error of the estimation is calculated according to equation 4.11, the error comes out to be 0.0734. The average estimation error is merely 7 centimeters.

### 4.5.2 Non-uniform Corridor

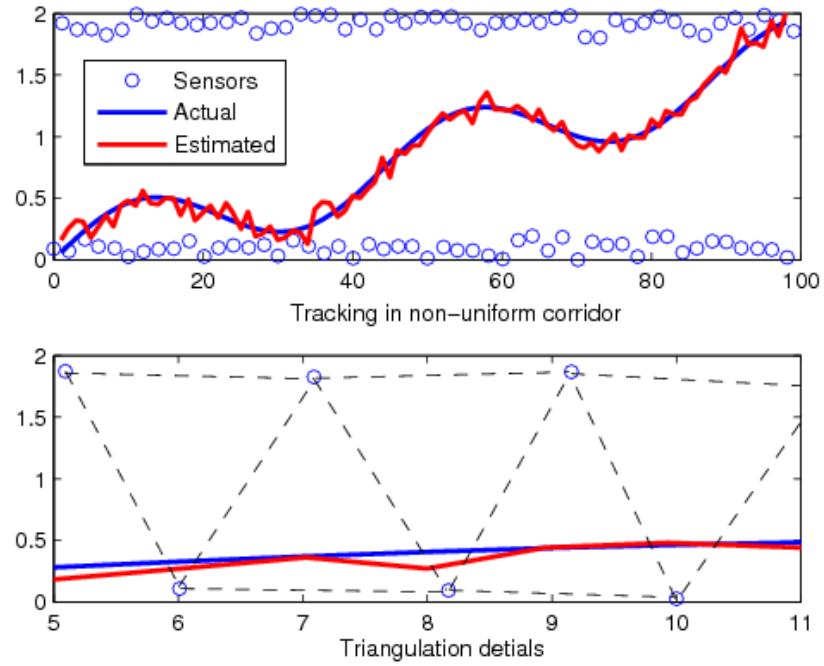
In the second, the same setup was used. The only difference is that the sensors were not equally spaced along the two boundaries. The sensor model and the target trajectory remains the same. Fig. 4.5(b) shows the tracking result. The estimation error come out to be 0.0748, which is approximately the same as the uniform corridor case. This further proves that under the clique tree structure, the non-uniform corridor behaves exactly the same ways as uniform corridor.

### 4.5.3 Random sensor field

In the randomly distributed sensor field, 100 sensors were randomly distributed in a 20-meter by 20-meter region. All other conditions were held exactly the same. In order to evaluate the robustness of the algorithm, there is a 10% failure chance applied to each sensor (i.e. for 10% of the time, the sensor will completely ignore the target). Two different noise level were applied, they were 0.1 and 0.5. The results are shown in Fig. 4.6, where Fig. 4.6(a) and Fig. 4.6(b) are the tracking results for the two different noise levels. The estimation error is calculated according to equation 4.11 and plotted in fig. 4.7.

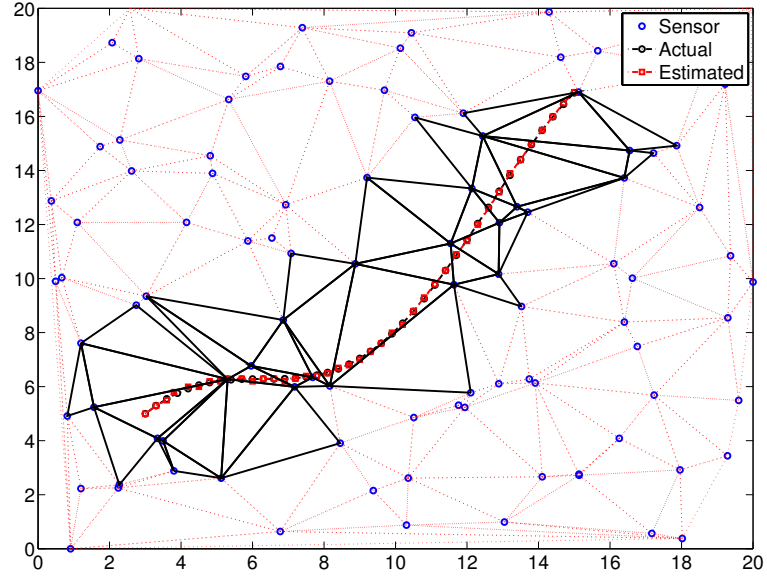


(a) Uniform distributed corridor

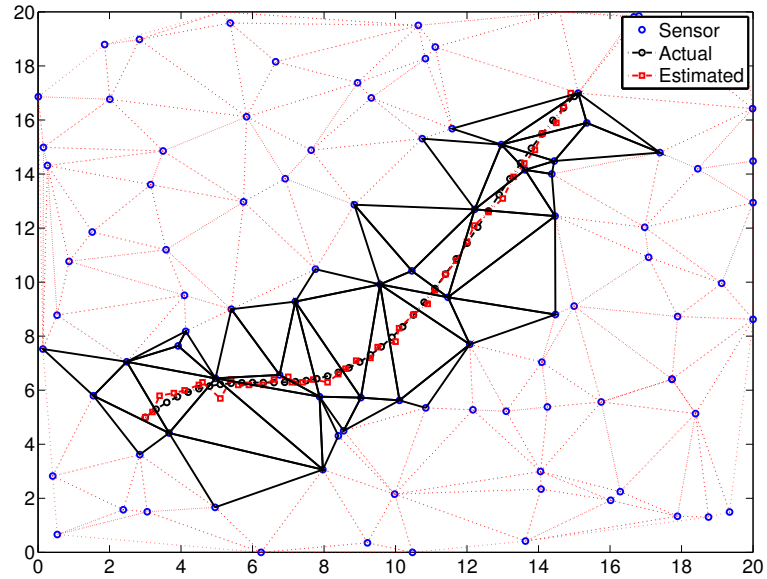


(b) Non-uniform distributed corridor

**Figure 4.5:** Tracking result for uniform and non-uniform chained-form regions



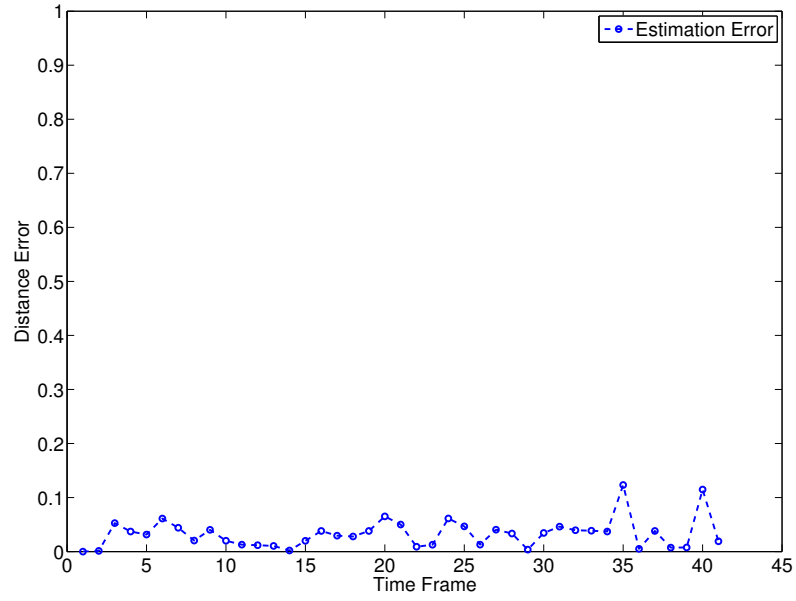
(a) Tracking result with low noise level,  $\sigma = 0.1$



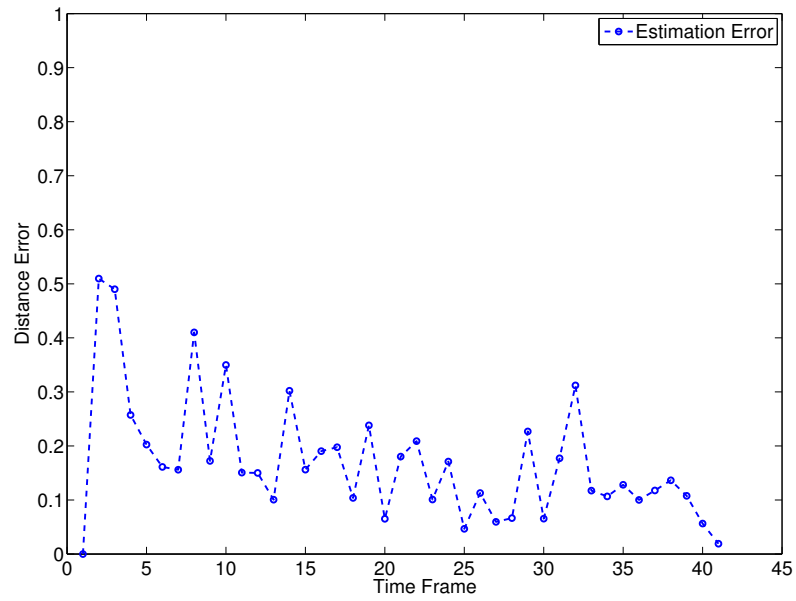
(b) Tracking result with high noise level,  $\sigma = 0.5$

**Figure 4.6:** Tracking result in random distributed networks and associated clique tree





(a) Estimation error with low noise level,  $\sigma = 0.1$



(b) Estimation error with high noise level,  $\sigma = 0.5$

**Figure 4.7:** Estimation error for random distributed sensor network

## 4.6 Conclusion

We have proposed a new distributed target tracking algorithm for sensor network using statistical graphical model. It addresses challenges including triangulate random distributed network into a chained-form, derive the clique tree and compute MAP estimation using message passing. Simulations have been conducted to study its performance. The algorithm proves to be less computational intensive, low cost and robust.

# Chapter 5

## Target Tracking with Grid-like Network Topology<sup>1</sup>

### 5.1 Introduction

Sensor network has found its way into many real world applications such as surveillance, traffic control, emergency response systems, and supply chain management systems. The core technology of these systems is the tracking algorithm, finding a good tracking algorithm not only enhances the tracking accuracy, lowers the computation cost, but also enables the system to perform target behavior analysis. There are usually two types of interested targets in tracking problems, the small target and the large target. The small target is usually assumed to be a small object that can be modeled as a point mass. When the small target is not moving, it is represented by a single Cartesian coordinate. When the target moves, it produces a continuous trajectory line. The large target is an area of events that cannot be considered as point mass, nor can it be accurately modeled by a coordinate. The area of events can be dynamic in size and location. The area of events is usually modeled by its boundary.

Sensor network has a distributed structure. At individual sensor level, the sensing and computation capabilities are limited. In order to make accurate inferences, sensors need to perform collaborative in-network processing. However, the sensors are usually deployed at

---

<sup>1</sup>©2009 IEEE. Portions reprinted with permission, from **Lufeng Shi**, and Jindong Tan, “*Distributive target tracking in sensor networks with a markov random field model*”, in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 854 - 859, 2009. See Appendix for a copy of the copyright permission from IEEE.

random, making it difficult to find an efficient collaboration pattern. The measurements of nearby sensors are usually correlated, making the collaboration event more difficult. The use of graphs made it possible to tackle these types of problems [16]. Graphs well capture the topology of the sensor network with vertices and edges, where vertices model the measurement uncertainties of individual sensors, and edges models the correlation between sensor measurements.

In most cases, sensor network are dense and large in scale, hence when events arise in the network, it is the best to keep the event local to a small group rather than evoke the whole network. This promotes the use of Markov random field (MRF). Markov random field is an advanced form of graphical model. In addition to simply model the uncertainty and correlation, the Markov random field models the conditional independency between nodes. The Markov random field property states that an event only concerns the detecting node and its immediate neighbors, and it is conditionally independent to the rest of the network. This isolates the events in the network for effective computation.

In this thesis, a distributed algorithm is developed for solving target tracking problems using Markov random field. The algorithm includes developing a universal technique that fits for both scenarios. The algorithm constructs a regularly distributed clique structure from the irregularly distributed sensor fields to provide easy path for collective data processing and search. The search engine is similar to the line search algorithm in image boundary detection, which is simply based on probability and likelihood rather than complex filtering.

## **5.2 Related Works**

In order to demonstrate our innovation, some of the past works regarding tracking and boundary detection are review for comparison purposes. The event boundary detection algorithms are discussed first, and then the boundary detection in image segmentations are reviewed.

There are some event boundary detection algorithms proposed for sensor networks that can be found in [30, 29, 27, 71, 28]. In these papers, the algorithms for detecting the boundaries are quite primitive. In [29, 71], the papers focused on fault-tolerant part of the algorithm, they used the event region detection process to identify the failing sensors, but the event region detection algorithm itself is a simple threshold based algorithm. Although in [27, 28] , the concept of probabilistic detection model is introduced, however, the determination rule at individual sensor is still threshold based. A staircase estimation method is used in [30]. It partitions the entire region into small cells, and collects the sensor measurements

located in each cell. The measurements in each cell is averaged, and feed into a piece wise constant estimator to find a straight line in that cell. The straight line computed is said to be an approximation of the boundary in that cell. This algorithm requires quite large overhead cost to setup, each sensors need to transmit at least  $\Omega(\log_2 n)$  to partition the cells. Once partitioned, each cell just acts on its own to compute the local estimates, thus the algorithm lacks of overall view of the entire network. Inferences in sensor network should be a collective effort among all sensors, hence the graph view of the network is the most appropriate method for solving sensor network problems.

If the topology of the sensor network is modeled as a graph, then the task of tracking is to locate objects in a graph. The concept of finding objects in graphs is not new. Images can be modeled by graphical models [39], and many algorithms have been developed to identify the objects in images. Once the image is modeled by a graph, many similarities can be identified between the image and the sensor network. Objects in images can be identified and detected by their boundaries. The boundary of an object in the image is essentially a number of pixels that separates the two types of texture pattern. In sensor networks, the sensors located at the boundary of an area of event have similar properties as these pixels. Hence, boundary detection algorithms in image segmentation can be adopted into sensor networks to solve target tracking problems.

There are many boundary detection algorithms in the image processing [72, 38, 73]. Generally speaking, the boundary detection algorithm is a learning process, and can be classified into supervised and unsupervised learning. In [38, 73], the paper addressed an unsupervised algorithm, in which the texture pattern of the background and the objects are unknown. It has to be discovered along with the boundary detection using empirical priors. The boundary detection problems in sensor networks, however, resemble more of a supervised learning problem. There are only two types of sensors (two known type of “texture”) in the network, they are either detecting the target, or not detecting the target. The supervised algorithm used in [72] is more appealing. A line search algorithm is discussed, where the boundary is merely a change point in the pixels, which separate two types of texture. Large target tracking in sensor network shows similar behaviors. The core concept of our algorithm is largely based on this approach.

This thesis proposed a graphical model based algorithm utilizing the existing line search algorithm in image segmentation. The algorithm is suitable for both small point-mass target and large event region detection scenarios. The algorithm makes decision based on probability distribution rather than threshold. The graph models the overall network connection, hence improves the collaborative data processing.

## 5.3 Problem Formulation

### 5.3.1 Graph Representation of Sensor Field

Consider a typical tracking problem in a sensor field. Assume in this sensor field  $S$  with  $K$  sensors, each sensor is capable of detecting the presence of the targets,

$$m_k = \begin{cases} 1 + n & \text{if target present } (H_1); \\ n & \text{if target absent } (H_0). \end{cases} \quad (5.1)$$

where  $m_k$  is a random variable that describes the detection status of the sensor  $k$ ,  $k \in \{1, 2, \dots, K\}$ , and  $n$  is zero mean noise. The distribution of the  $m_k$  is dependent on the noise distribution,

$$\begin{aligned} P(m_k|H_0) &= P_n(m_k) \\ P(m_k|H_1) &= P_n(m_k - 1), \end{aligned} \quad (5.2)$$

where  $P_n(m_k)$  is the noise distribution.

Given the sensing model of the individual sensor, the sensor field can be modeled using a graph. In the graph  $G = (V, E)$ ,  $V$  represent the  $K$  sensors in the network, and  $E$  represent the measurement correlations between sensors. Each  $m_k$  corresponds to a measurement in  $V_k$ . According to the detection model of individual sensors, any two sensors that are within each other's detection range are correlated, and this makes the statistical information in the network redundant and difficult to analyze, as shown in Fig. 5.1(a). To reduce the statistical redundancy, the sensor field is assumed to be a Markov random field, where edges (correlations) only exist between immediate neighbors, as shown in Fig. 5.1(b). Even when two sensors are in range of each other, if they are separated by other sensors, they are conditionally independent to each other given the middle sensors. The conditional independency can be expressed mathematically as

$$\prod_{k=1}^K p(m_k|m_1, m_2, \dots, m_K \setminus m_k) = \prod_{k=1}^K p(m_k|NB(m_k)), \quad (5.3)$$

where  $NB(m_k)$  represent the immediate neighbors of  $k$ . Each connected neighborhood can

be called a clique. The cliques are independent of each other, and hence the joint probability density distribution can be computed by multiplication. Let  $M = \{m_1, m_2, \dots, m_K\}$ , the joint distribution of the sensor field is modeled using the Gibbs distribution,

$$p(\mathbf{M}) = \frac{1}{Z} \exp \left( \sum_{c \in C} (\psi_c(m_k)) \right), k \in c \quad (5.4)$$

where  $c \in C$  are the cliques in the field, and  $\psi_c(m_k)$  represent the clique potential.  $Z$  is a normalization constant. With this network model, when computing the joint probability distribution, the probability distribution of each clique can be pre-computed locally, and hence it is distributed in-network processing.

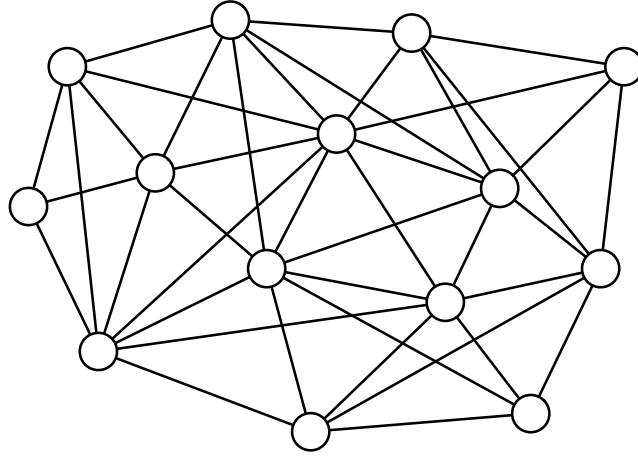
### 5.3.2 Definition of Targets

In this thesis, the targets in the sensor network can be classified into two categories, the small targets and the large targets. The small target can be modeled as a point mass. A fixed small target is a single Cartesian coordinate  $(x_t, y_t)$ . This allows us to do numerical manipulations such as measure the distance and perform multilateration. Once the target starts moving, it produces a continuous trajectory line. The usual way of modeling this trajectory line is to capture a static snapshot of the coordinate at each time frame, and the trajectory line can be represented by a vector of coordinates  $\theta$ . Assume the sample is taken over  $T$  time period,

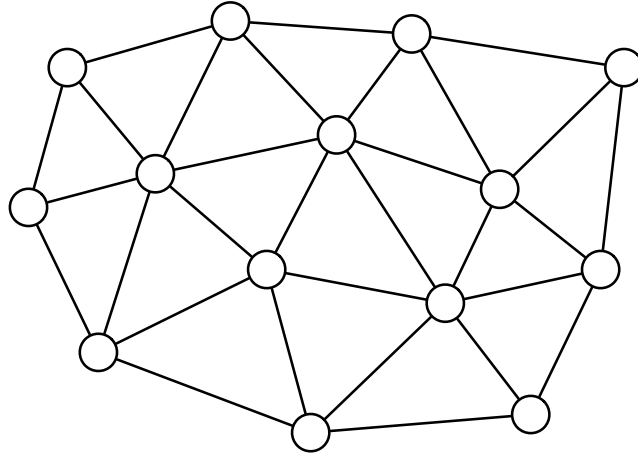
$$\theta = \{(x_t^1, y_t^1), (x_t^2, y_t^2), \dots, (x_t^T, y_t^T)\}. \quad (5.5)$$

As shown in Fig. 5.2(a), black squares are the snapshot coordinates at each time frame, the orange line is the target trajectory, and the circles are the sensors. The connected red circles are the sensors that are most relevant to the target trajectory; they will form a chained-form network which is used to estimate the location of the target and provide a routing path for data aggregation. The challenge is to identify these sensors from a randomly distributed network as shown in Fig. 5.1(b).

A large target is an area of event, where a single Cartesian coordinate cannot be used to represent this type of target, hence we cannot obtain a collection of coordinates over time to represent the large targets. The boundary of a large target is a continuous line, which exhibits a similar characteristic as the trajectory of the small target, thus a chained-form network can be identified as shown in Fig. 5.2(b) to represent the current location of the target. The main objective for tracking both large targets and small targets is to identify the chained-form network as shown in Fig. 5.2.



(a) Network topology with all possible edges shown



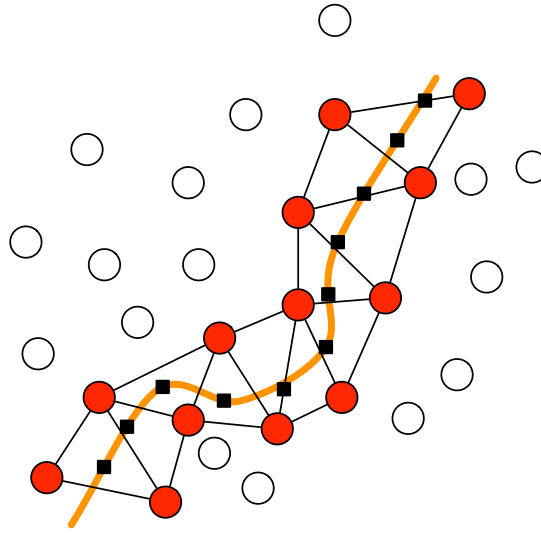
(b) Network topology after assuming Markov property

**Figure 5.1:** Graph representation and Markov random field representation of the same sensor field.

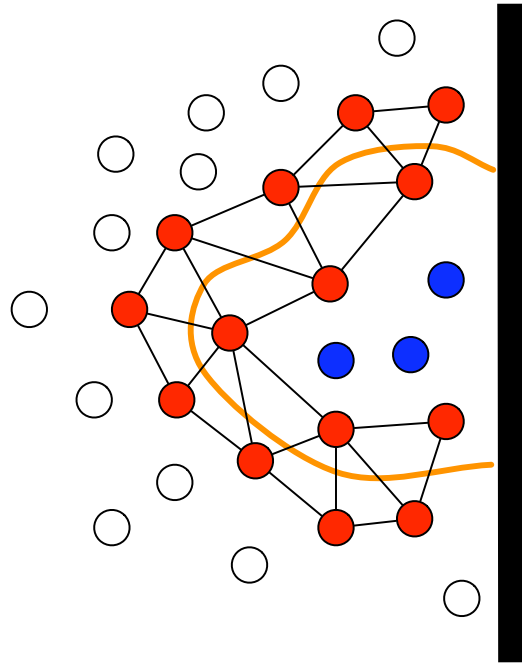
## 5.4 Tracking in A Graph With Ising Model

The Ising model is a Markov random field with a grid topology. With this regular distributed network, it is much simpler to identify the chained-form network. Fig. 5.3 shows a possible target trajectory (boundary) in Ising model, where the circles are the sensors. The hollow circles represent the null detection sensors, and the colored circles represent sensors that are detecting the target. The goal is to identify the boundary of the





(a) Representation of a small target in sensor networks



(b) Representation of a large target in sensor network

**Figure 5.2:** Similar representation of various sized targets in the sensor network.



before  $H_1$  in the line of sensors,

$$\begin{array}{ll} m_1^\delta & \text{in state } H_0 \\ m_\delta^l & \text{in state } H_1 \end{array} \quad (5.7)$$

According to the Markov property, the probability of having  $\delta$  as change point can be expanded as

$$P(\delta \in \text{boundary} | m_1^l, H_0, H_1) = CP(m_1^\delta | H_0)P(m_\delta^l | H_1), \quad (5.8)$$

where  $C$  is a normalization constant.

With equation (5.8), we can search through the all possible  $\delta$  to find the most probable boundary location

$$\delta = \arg \max_{\delta} P(\delta \in \text{boundary} | m_1^l, H_0, H_1). \quad (5.9)$$

Equation (5.8) is only valid if the segment  $m_1^\delta$  is in state to  $H_0$ , and segment  $m_\delta^l$  is in state  $H_1$ . This is because the term  $P(m_1^\delta | H_0)$  pares  $m_1^\delta$  with state  $H_0$ . In real applications this is not always the case. Sensors  $m_1^\delta$  may be in either state  $H_0$  or  $H_1$ , hence  $P(m_1^\delta)$  should be used in place of  $P(m_1^\delta | H_0)$ , and it can be computed using the prior,

$$P(m_1^\delta) = \int P(m_1^\delta | H)P(H)dH \quad (5.10)$$

where  $P(H)$  is the joint prior of  $H_0$  and  $H_1$ .

The assumption for Ising model is essential. Without the grid shape, the search will encounter unexpected branches and the correlation will vary from sensor pairs to sensor pairs. This will cause additional difficulty in performing the search. However, in typical sensor network deployment, networks are usually not grids. To make this algorithm applicable to all kinds of sensor networks, an equivalent topology structure for the sensor network must be derived which must satisfy the grid assumption. In this thesis, a clique based topology representation is used to convert the irregular topology of the sensor network to a grid-like structure, then the algorithm is applied to this grid-like structure.

## 5.5 Tracking in Randomly Distributed Sensor Network

Typical sensor networks are randomly distributed, and the Ising model assumption cannot be achieved. However, by exploit the property of cliques and conditional independence, the randomly distributed network can be converted into more predictable shapes. Therefore, many algorithms developed for regularly distributed network can be extended to a randomly distributed network with little modification.

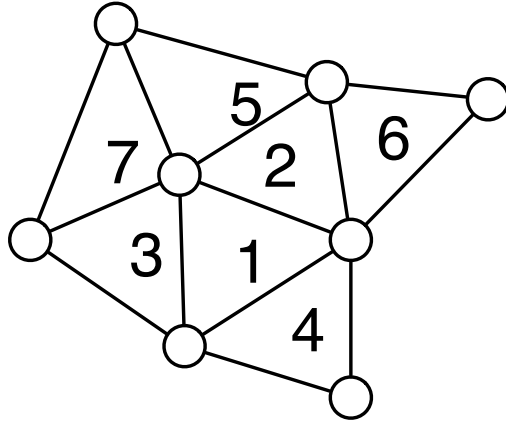
### 5.5.1 Grid-like Topology Construction

In an Ising model, each node has exactly four connected neighbors (except borders and corners), and the distance (correlation) between each neighbor is exactly the same. In order to construct a topology structure similar to the Ising model, we have to satisfy these two conditions: 1) have a fixed number of connected neighbors, 2) have the same distance (correlation) between connected neighbors.

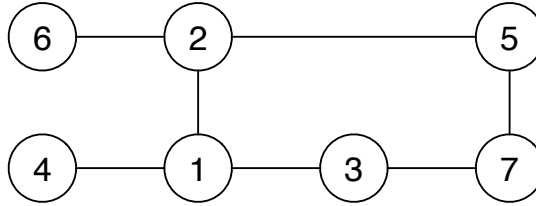
These two conditions can be easily satisfied by adopting triangular cliques. A clique is a cluster of sensors which are fully connected within the cluster. When three node triangle cliques are used, each clique would have exactly three adjacent triangles. The adjacent cliques are correlated by the two shared nodes. Hence, the clique structure can be used to simulate the behavior of a grid structure. Fig. 5.4 shows how a randomly distributed sensor topology can be converted into a grid-like structure. The network is first triangulated, each triangle is a natural clique, then a single “super node” is used to represent a whole clique in the grid-like structure.

### 5.5.2 Void Area in Grid-like Topology

The most popular triangulation algorithm is the Delaunay triangulation. However, Delaunay triangulation cannot be applied to the sensor network directly. This is because the algorithm does not have a bound on edge length; it may result in long edges that far exceeding the actual sensing and communication range of the sensors. At the same time, Delaunay triangulation is very difficult to achieve in a distributed fashion, hence it is not suitable for distributed sensor network. Therefore, a sub-graph of the Delaunay triangulation such as Relative Neighborhood Graph (RNG), or Gabriel Graph (GG) can be obtained. However, in RNG or GG, polygons may exist, where three-sensor cliques cannot



(a) Random deployed sensor field

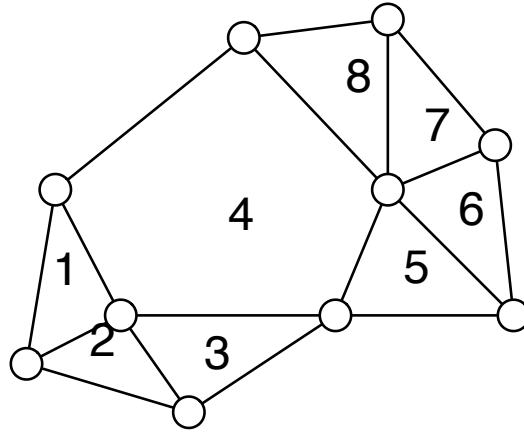


(b) Grid-like structured clique topology

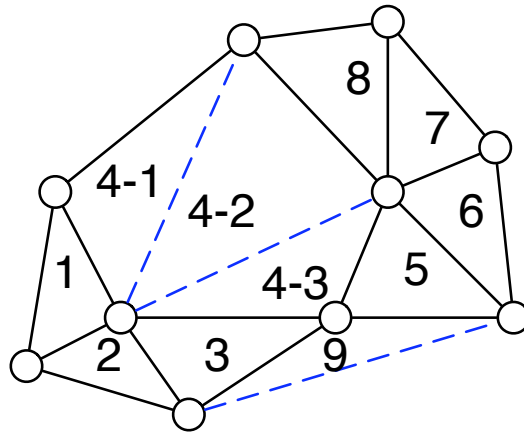
**Figure 5.4:** Converting the randomly distributed sensor network into a grid-like clique structure

be constructed as shown in Fig. 5.5(a). Area 4 in the figure is a void that cannot be modeled as a triangle clique.

Double-sensor cliques are introduced to solve this problem. For example, in Fig. 5.5(a), the five edges on the pentagon are modeled as five double-sensor cliques. Two of these double-sensor cliques can join together to form a virtual triangle if they share one sensor. A virtual link is added between the two non-sharing sensors in the group to finish the triangle. Fig. 5.5(b) is a demonstration of this situation. The dashed lines are virtual lines added to divide the polygon region into virtual triangles. Triangle 4 – 2 is a special case, where it is formed by only one double-sensor clique. So, 4 – 2 is treated as a triangle when constructing the clique structure, but when computing the joint distribution, it is just a single double-sensor clique. The construction of the virtual triangles obeys the following five rules as described in Algorithm 2.



(a) Triangulated sensor field with coverage holes



(b) Virtual triangles constructed to get across void areas

**Figure 5.5:** Converting the randomly distributed sensor network into a grid-like clique structure

### 5.5.3 The Tracking Algorithm in A Grid-like Structure

After the grid-like structure is constructed, the search algorithm for Ising model can be applied to identify the trajectory (boundary) of the target. However, since we don't have a true grid structure, and each "node" in our topology actually represents three nodes in the original structure, the detection models need to be modified slightly. The new detection

---

**Algorithm 2** Constructing virtual triangles with double-sensor cliques

---

1. An edge is a double-sensor clique if at least one of its adjacent triangles is missing.
  2. In order to form a virtual triangle, the two real edges must both be double-sensor cliques.
  3. Virtual links can be treated as double-sensor cliques when constructing virtual triangles,
  4. Virtual links are not included when computing the clique potential and other statistic related quantities.
  5. Virtual triangles can be formed if the added virtual link does not cross any of the existing links (virtual or real).
- 

model can be expressed as

$$\begin{aligned}
 P(H_0) &= \prod_{k=1}^3 P(m_k = n) = \prod_{k=1}^3 P_n(m_k) \\
 P(H_1) &= \prod_{k=1}^3 P(m_k = 1 + n) = \prod_{k=1}^3 P_n(m_k - 1),
 \end{aligned} \tag{5.11}$$

With  $P(H_0)$  and  $P(H_1)$  computed, the  $P(C_1^\delta)$  and  $P(C_\delta^l)$  can be computed. Note that  $l$  is the total number of the cliques in the line,  $\delta$  is the change point, and  $C$  is the cliques. Since computation of  $P(C_1^\delta)$  and  $P(C_\delta^l)$  are symmetric, we only focus on the computation for  $P(C_1^\delta)$ .

Given the detection result, the readings of each clique is independent to each other, hence,

$$P(C_1^\delta | H) = \prod_i P(C_i | H). \tag{5.12}$$

We can then expand equation (5.10) as,

$$\begin{aligned}
P(C_1^\delta) &= \int P(C_1^\delta|H)P(H)dH \\
&= \int P(C_\delta|H)P(C_1^{\delta-1}|H)P(H)dH \\
&= P(C_1^{\delta-1}) \int P(C_\delta|H)P(H|C_1^{\delta-1})dH.
\end{aligned} \tag{5.13}$$

This separates the terms containing the current clique  $C_\delta$  and all previous cliques  $C_1^{\delta-1}$ , hence it can be carried out in a distributed fashion using message passing algorithm. Each clique need only to compute their local  $P(C_1^\delta)$  and pass it down the line; the next clique in line will treat the received value as  $P(C_1^{\delta-1})$ . With the detection pattern defined, the equation 5.8 can be rewritten as

$$\begin{aligned}
P(\delta \in \text{boundary} | C_1^l, H_0, H_1) & \\
&= KP(C_1^\delta)P(C_\delta^l) \\
&= KP(C_1^{\delta-1}) \int P(C_\delta|H)P(H|C_1^{\delta-1})dH \\
&\times P(C_\delta^{n-1}) \int P(C_\delta|H)P(H|C_\delta^{n-1})dH
\end{aligned} \tag{5.14}$$

This equation works if there is only one change point  $\delta$  in the line of cliques, however, there may be several change points in each line.

Let us introduce a new variable  $\mu$  to represent the number of change points in the path, and tweak the equation a little to incorporate this variable,

$$\begin{aligned}
&P(\delta \in \text{boundary}, \mu | C_1^l, H_0, H_1) \\
&= P(\mu)P(\delta \in \text{boundary} | \mu, C_1^l, H_0, H_1), \\
&= P(\mu)KP(C_1^{\delta_1}) \prod_{i=1}^{\mu-1} P(C_{\delta_i}^{\delta_{i+1}})P(C_{\delta_\mu}^l)
\end{aligned} \tag{5.15}$$

$P(\mu)$  is a priori distribution; it can be obtained empirically. For instance, if a large target is known to be a circular shape, a Poisson distribution with expected value of 2 should be a good assumption, because a straight line of sensors would have two intersections with the



circle.

## 5.6 Simulations

For the purpose of demonstration, assume 100 sensors are deployed in a 20 by 20 square region. Each sensor has a sensing radius of 5. The algorithm is carried out on this square region to detect a small target, a large target, and multiple large targets.

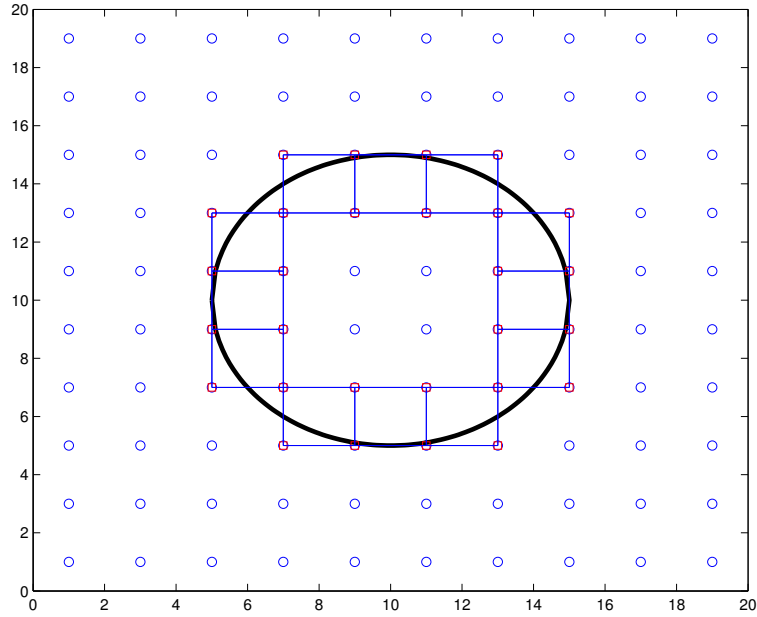
The search is first conducted on a sensor field that is modeled by an Ising model (grid). Fig. 5.6 shows the results. Since it is a grid network, no triangle clique is constructed. Circles are sensors, the red ones are the sensors that detecting the large target. The lines are the links in the chained-form network representing the target boundary.

For the randomly deployed sensor field, a Gabriel Graph is constructed on top of the sensor field, then virtual lines are constructed to break the polygons into triangle cliques. The triangle cliques are represented using “super nodes”, and the “super nodes” topology is arranged into a grid-like structure.

Fig. 5.7 shows the detection of the large target in the randomly deployed sensor field. The red triangles are real triangle cliques, and the blue lines are virtual lines that connect the double-sensor cliques into virtual triangles. Only the cliques that are representing the boundary of the target are shown.

Since the search algorithm is carried out on a Markov random field, the detection of the target is independent given the immediate neighboring sensors. Hence, if we have multiple large targets, they can be tracked as well, using the same search algorithm. Fig. 5.8 shows the detection of two large targets in the same sensor field.

Finally, small target trajectory is searched by the algorithm, and located on the sensor field with a chained-form clique. Fig. 5.9 shows the tracking result for the small target. Once the chained-form network is constructed, multilateration can be used in each triangle cliques to compute the exact position of the target at each time instance. The formulation of the multilateration can be found in [74].



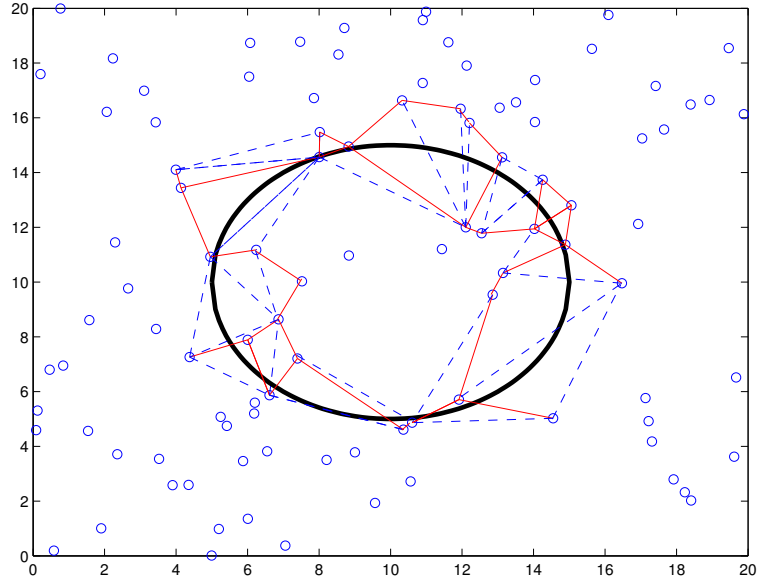
**Figure 5.6:** Large target in a grid sensor field using Ising model

## 5.6.1 Cost Contrast

When the grid-like network is setup using the “super” node, many algorithms that are impossible to implement on random network can now be utilized. In addition to provide a simpler structure, the grid-like network reduces the communication cost and computation cost for propagate message through the network, and it also provide self healing against node failure, improve overall robustness of the system.

### 5.6.1.1 Communication Cost

The most intuitive, most primitive communication algorithm in random sensor network is flooding algorithm. However, flooding algorithm usually induces high communication and computation cost. This is due to several shortcomings for the flooding algorithm. The first shortcoming is that it may never converge if there are loops in the network topology. This shortcoming can be avoided by setting termination condition, and add extra fields in the transmitted message indicate the message sequence and life span. However, the side effects are increase message size, extra computation upon reception of the message. The



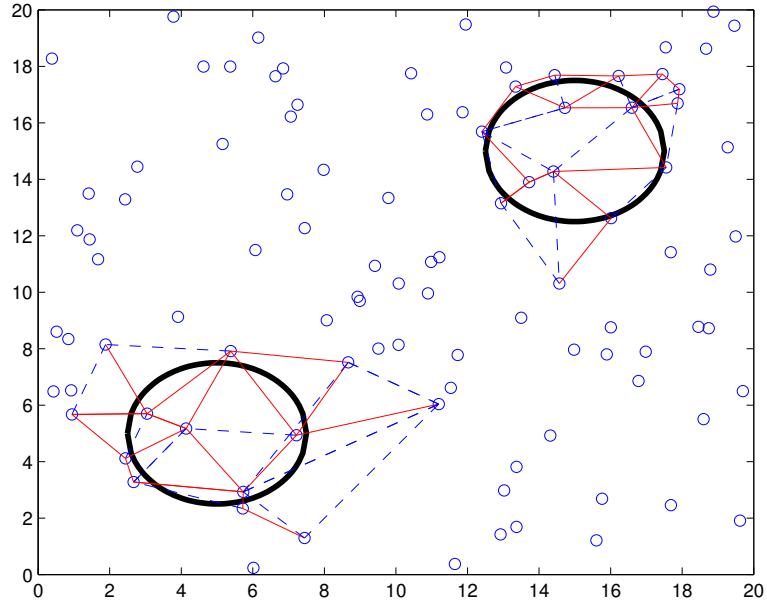
**Figure 5.7:** Large target in a randomly deployed sensor field

other shortcoming is that the flooding algorithm has the highest worst case complexity, in addition to the worst case complexity, the complexity in actual use case varies from situation to situation, which adds additional uncertainty to the system.

After the concept of wireless sensor network flourished, many works have been done on tweaking the classic flooding algorithm to work better in the world of sensor networks [75] [76] [77]. There are many that claim to reduce the complexity of the flooding algorithm for sensor networks, but most of them are just setting up intelligent control conditions to terminate the flooding [78] [79] [80] [81]. By doing so, flooding can be simplified by reducing by sacrificing some of the non-critical information. However, even with the controlled flooding, the best complexity it flooding can achieve is still  $O(n \log n)$ .

The grid-like network greatly reduce the uncertainty in the sensor network, message propagate through the network in a controlled fashion, hence keeping the logic simple, the communication and computation cost low. In this section, we compare the cost of the flooding algorithm and the grid-like network back-to-back to evaluate the potential of the grid-like network. In here, we consider two common scenarios in target tracking in sensor networks.

Let us first consider a situation where a sensor generated a reading that needed to notify the entire network. With flooding algorithm, the worst case scenario, the message need

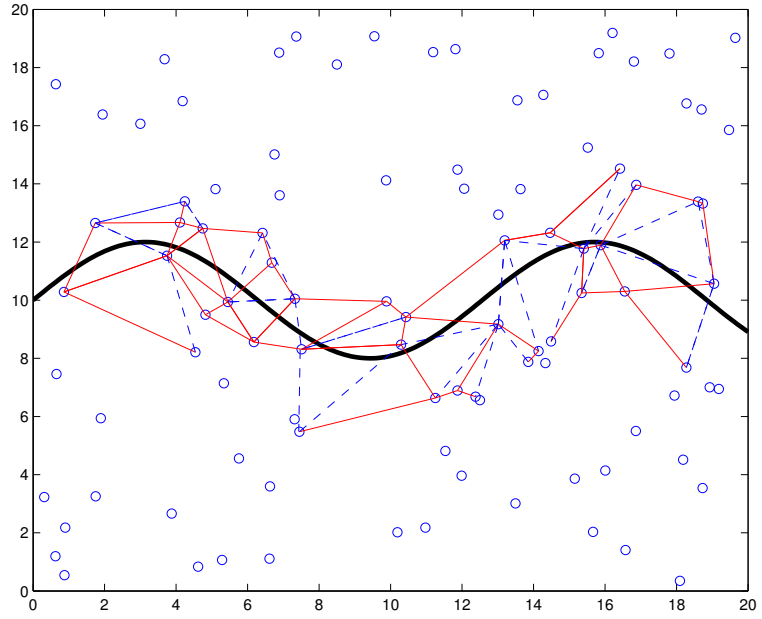


**Figure 5.8:** Detection of two large targets in the sensor network

to be transmitted over each of the link in the network at least once. In a fully connected network, the number of links in the network can be calculated by  $\frac{n(n-1)}{2}$ , where  $n$  is the number of sensor nodes in the network. Hence the number of message to be transmitted is proportional to  $n^2$ . In grid-like network structure, after the network is constructed, we only need to visit each sensor node exactly once, hence giving us a flat  $O(n)$  complexity, where  $n$  message is to be transmitted. The figure 5.10 below shows the number of message needed to be sent by flooding algorithm and by the grid-like network in order to announce the data to all the nodes in the network, assuming there are 100 sensor nodes in the network. For grid-like method, we assume overhead cost of 10000 messages for setting up the grid-like structure ( $O(n^2)$ ).

The second scenario to consider is that when a sensor is currently detecting the target, and it is trying to exchange data with nearby sensors in order to achieve collective inference. In flooding algorithm, the worst case would be the data from the host sensor need to take  $O(n^2)$  messages to reach the targeted sensors. For grid-like network, to inform all neighbors, each “super” node only need to inform its 4 connected neighbors. Hence between “super” nodes, messages exchanged is 4.

As shown in the contrast, the grid-like network is definitely showing a big communication cost reduction over the flooding algorithm. However, flooding algorithm has its own strength. The flooding algorithm so far is the only true infrastructure-less algorithm that

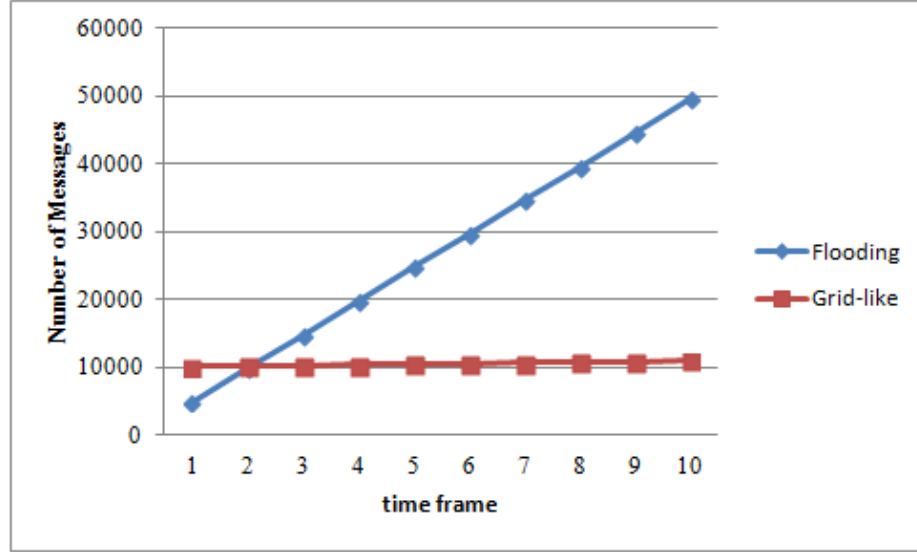


**Figure 5.9:** Small target trajectory in the sensor field

**Table 5.1**  
Number of messages sent by each method

Time	Flooding	Grid-like
1	4950	10100
2	9900	10200
3	14850	10300
4	19800	10400
5	24700	10500
6	29700	10600
7	34600	10700
8	39600	10800
9	44550	10900
10	49500	11000

will accommodate any network topology change. If some sensor nodes fail, or move, the flooding algorithm is practically unaffected by them at all. However, most controlled topology like grid network will take a big hit when topology change occurs. In this section, the self-healing possibility is also discussed to accommodate for potential cost increase due to sensor failure.



**Figure 5.10:** Number of Messages sent by each method

### 5.6.1.2 Self healing

Sensor failure is common in sensor network due to the fundamental assumption of wireless sensor network. The individual sensor supposed to be “low cost”, “less powerful”, “limited power”. So individual sensor can easily fail due to loss of power, moved out of the designated location due to environment change, or simple due to circuitry imperfection. When these failures happen, flooding algorithm is practically unaffected by these change, but for grid network, this will cause possible coverage hole, communication link loss, which may eventually cause a huge failure in the entire network.

The grid-like network proposed in this thesis however is quite robust against sensor failures. In case of individual sensor failure, since the “super” nodes are conditionally independent to each other, the change is only confined within the “super” node, hence have very small impact on the rest of the network. In case that the entire “super” node fails (i.e. the area that the “super” node resides is destroyed entirely), the “super” node just simply detached from the rest of the network, without affect the rest of the network. The cost of recover from an individual sensor failure is  $O(n_c)$ , where  $n_c$  is the number of the sensors within the “super” node, the rest of the sensors in the “super” node just has to realize the failed sensor is been detached from the clique.

## 5.7 Conclusion

A distributed algorithm is developed to solve target tracking problems in sensor network. The algorithm can be used to find both the trajectory of the small targets and the event boundary of the large targets. The core concept of the algorithm is a simple search method inherited from the line search algorithm in image segmentation. The new algorithm is distributed which allows large scale implementation. It is also probability based, where the decision is based on probabilistic distribution, detection errors of individual sensors affect little to the overall result. Simulations demonstrate the functionality of the algorithm against small target, large target, and multiple targets.

## **Chapter 6**

# **Self Localization of Ultrasonic Sensor Network**

### **6.1 Introduction**

Ultrasonic sensor arrays have been widely used in many applications, where precise localization is needed [82] [83] [84]. Recent combination of ultrasonic sensors and wireless sensor networks provides a new and unique platform for distributed target localization and tracking. In an ultrasonic sensor network, the individual ultrasonic sensors are linked together wirelessly instead of mounted on the same circuit board, this distributed structure provides more flexibility in sensor deployment, hence allowing a much larger coverage area than traditional ultrasonic sensor arrays. With this distributed architecture, individual ultrasonic sensor failures will have little impact on the overall system.

There are many challenges in developing applications using ultrasonic sensor networks. Energy management is one of the major challenges. Ultrasonic transducers require large amount energy to operate, they will squeeze the already limited power supplies of wireless sensor networks. In addition, to computing the time difference of arrival, more communication will be needed, inducing more energy expenditure. Hence energy management is required to achieve the pre-longed network service time. Sensor localization is also very important in ultrasonic sensor network as well. Ultrasonic sensor based localization systems require the knowledge of the exact location of the sensor itself in order to produce high precision tracking, but in a randomly distributed sensor network, the sensor locations are usually not known to begin with. In the end, since the sensors are randomly deployed, the coverage and interference of the ultrasonic sensors can be



a problem. Optimization is needed to maximize the coverage area while minimize the interference.

This thesis focuses on the challenges of localization, energy, and coverage and interference management of the ultrasonic sensor networks. A feasible localization technique for randomly distributed ultrasonic sensor network is provided. Then a scheduling technique is discussed to optimize the energy consumption, enlarge the sensor coverage region, and minimize the ultrasonic over hearing and interference. Since our application for the ultrasonic sensor network is tracking and localization of mobile target, the sensor scheduling technique is presented in two difference scenarios, when targets are presenting, and when no target is presenting. It is shown that the target will help sensor scheduling even if the target is non-cooperative.

## **6.2 Related Works**

There many prior researches regarding localization in wireless sensor networks [85] [86]. However, there are a few limitations on these techniques that prevent them to be successfully applied to an ultrasonic sensor network. These localization techniques assume a number of anchor nodes with pre-known location information, and the anchor nodes are able to measure the distance to other unknown sensors. This is not realistic in ultrasonic sensor network. The ultrasonic sensors does not have an omni-directional measurement ability, and the static ultrasonic sensors are typically assumed not be able to rotate. Hence, the anchor node would not be able to measure the distance to the unknown sensors that it is not facing. For ultrasonic sensor network, a new localization technique is needed.

Ultrasonic sensor networks has been used in falling detection in medical facilities [87] [83]. In these types of applications, instead of having ultrasonic transmitter and receiver mounted on each sensor node, the transmitter is unique and is carried by the patient, and the receivers are networked together. In such a situation, the pose of the patient can be estimated using the time difference of arrival to the receivers. These types of ultrasonic sensor networks are much easier to work with since the source is unique, interferences will not be a problem. Multi-path maybe a problem, but it is not addressed in both of the papers.

Energy consumption and coverage optimization is discussed in [88]. In this paper, the author formulated the optimization of the energy consumption and coverage. Although there are multiple transmitters presenting, the paper still assume that only one sensor can be turned on at one time to avoid interference. The paper presented a sensor scheduling method that strategically switch on receivers to maximize the coverage range with minimum number of receivers, hence achieve both good coverage region and conserve

energy.

Target tracking problems are discussed in [84] [89] [90]. These papers all utilize the Kalman filter and its variants to estimate the mobile target within the ultrasonic sensor networks. However, despite the distributed claims, the sensors are still pre-clustered, and arranged into array like structure for processing before tracking the targets. The papers are primarily focused on hardware development, the algorithms used are standard Kalman filter technique, where interference and coverage problems are not specifically considered. Multiple targets situation is not considered as well.

The contribution of this thesis is to develop a framework for randomly deployed ultrasonic sensor network. The framework uses simple ultrasonic sensor nodes, where each node only has one source and one receiver on board. The framework handles the challenge in localization, it provides a scheduling scheme that allows multiple unaffected ultrasonic sources to switch on simultaneously to avoid interferences, and it can schedule the sensors to actively switch on along the trajectory of the mobile targets.

## 6.3 Self-localization

Self-localization is a basic middleware service for any wireless sensor networks. The ultrasonic sensor network can only providing tracking services with all the sensors localized. Typical self localization of sensor network is usually based on two assumptions. A first assumption is that there are anchor nodes with pre-known location information. The second assumption is that the sensors could measure the distance between each other. Both of these assumptions are difficult to achieve in ultrasonic sensor networks. The anchor nodes are usually assumed to be equipped with GPS, hence are able to determine their locations. In an in-door environment, where the ultrasonic sensor networks are deployed, GPS will not be a feasible option. The ultrasonic sensors are not omni-directional, the facing directions of the ultrasonic sensors dictate where they could measure, hence assuming the sensors could measure the distance to all the neighbors is also not a valid assumption.

In this thesis, a triangular trainer is used to help localize the sensors. The triangular trainer has three vertices, the ultrasonic sensors could measure the distance to each of the vertices independently. If the trainer's location is known, then sensors that detecting the trainer can be localized using tri-lateration technique. The sensors are placed in a random fashion, distances from the sensor to each of the vertices of the triangle are measured to be  $L_{1t1}$ ,  $L_{1t2}$  and  $L_{1t3}$ . Assume the location of the three vertices of the trainer to be  $(x_{t1}, y_{t1})$ ,  $(x_{t2}, y_{t2})$

and  $(x_{t3}, y_{t3})$ , the target location  $(x_1, y_1)$ , can be formulated as,

$$\begin{aligned}(x_{t1} - x_1)^2 + (y_{t1} - y_1)^2 &= L_{1t1}^2 \\(x_{t2} - x_1)^2 + (y_{t2} - y_1)^2 &= L_{1t2}^2 \\(x_{t3} - x_1)^2 + (y_{t3} - y_1)^2 &= L_{1t3}^2\end{aligned}\tag{6.1}$$

where  $(x_1, y_1)$  is the location of sensor 1 that is to be estimated. Expand the equations to get

$$\begin{aligned}x_{t1}^2 + x_1^2 - 2x_{t1}x_1 + y_{t1}^2 + y_1^2 - 2y_{t1}y_1 &= L_{1t1}^2 \\x_{t2}^2 + x_1^2 - 2x_{t2}x_1 + y_{t2}^2 + y_1^2 - 2y_{t2}y_1 &= L_{1t2}^2 \\x_{t3}^2 + x_1^2 - 2x_{t3}x_1 + y_{t3}^2 + y_1^2 - 2y_{t3}y_1 &= L_{1t3}^2\end{aligned}\tag{6.2}$$

If we subtract the top two equations from the bottom equation, we can obtain a set of linear equations

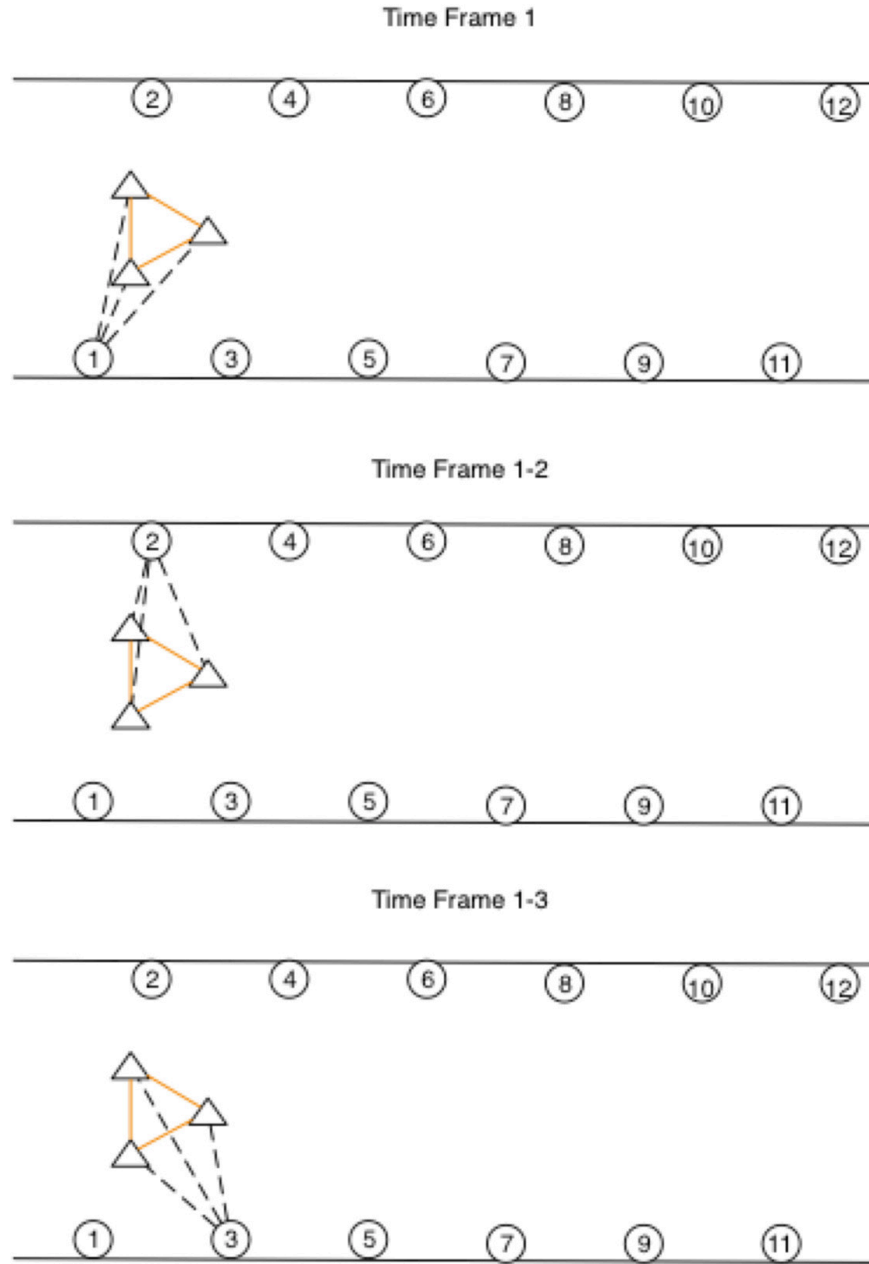
$$\begin{aligned}&2(x_{t3} - x_{t1})x_1 + 2(y_{t3} - y_{t1})y_1 \\&= x_{t3}^2 - x_{t1}^2 + y_{t3}^2 - y_{t1}^2 - L_{1t3}^2 + L_{1t1}^2 \\&2(x_{t3} - x_{t2})x_1 + 2(y_{t3} - y_{t2})y_1 \\&= x_{t3}^2 - x_{t2}^2 + y_{t3}^2 - y_{t2}^2 - L_{1t3}^2 + L_{1t2}^2\end{aligned}\tag{6.3}$$

Setup the linear equation in standard  $Au = \beta$  form, we could compute  $u$  using the least mean square method

$$u = (A^T A)^{-1} A^T \beta,\tag{6.4}$$

$u$  will consist the coordinates of the sensor. The localization process is demonstrated in Fig 6.1, where the sensors are deployed in a hallway.

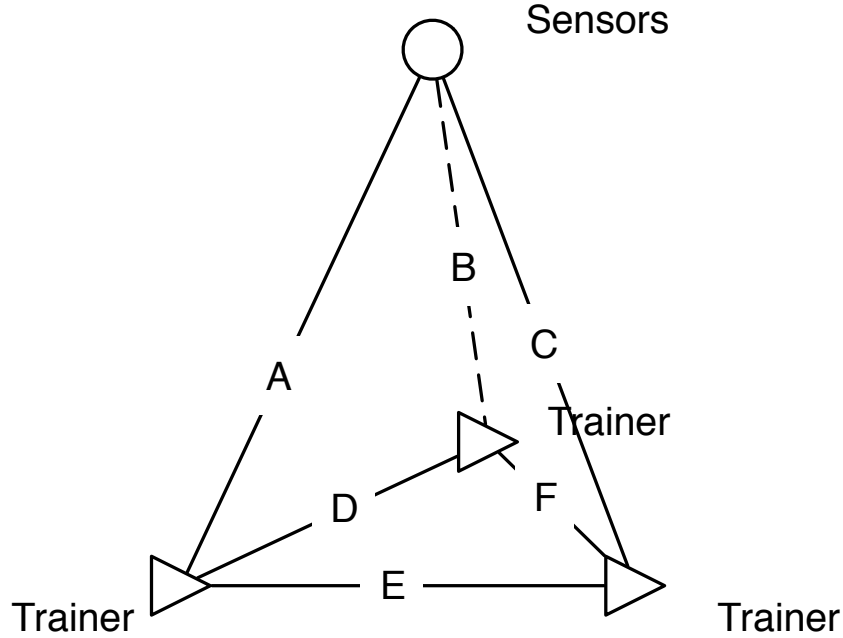
As shown in the figure, not all the sensors are in range with the trainer, hence the trainer must travel through the sensor field in order to reach more sensors. Once the trainer moves, we need to determine its new location. We assume the trainer will move at a relatively constant speed, and measure this speed using UWB sensors that are integrated with the ultrasonic sensor. The new location of the target is estimated using this measured speed and its previous location.



**Figure 6.1:** The estimation process of each sensor in a hallway, given the trainer location

In this thesis, we primarily discuss two types of sensor network settings, the first setting is a hallway setting as shown in Fig 6.1, the other setting is a room setting where an open space is available, and the sensors are deployed randomly on the ceiling facing downward. In such a 3D situation, in order to estimate the sensor location, we first find the projection

the sensor location on the ground level using tri-lateration, then find the original location of the sensor using the height of the ceiling. The sensors in the ceiling and the triangular trainer forms a tetrahedron as shown in Fig 6.2, in this case, the height of the ceiling can be computed using  $h = 3V/b$ , where  $V$  is the volume of the tetrahedron, computed using Tartaglia's formula, and  $b$  is the area of the base triangle, which is essentially the area of the trainer. Once we determined the height of the ceiling, we could project the measured distance from the sensor to each of the vertices on the trainer to the same plane as the trainer, and then perform tri-lateration.



**Figure 6.2:** Tetrahedron formed with the sensor and the triangular trainer

Since the self-localization requires a trainer traverse through the sensor field, we could use it to perform a Reference Broadcasting Synchronization (RBS) [91] at the same time. In RBS synchronization, a trainer broadcast a periodic synchronization beacon, and the ultrasonic sensors will adjust their clock to the same pace as the trainer. In this way, the critical path on the sender side can be completely eliminated. Since the self-localization is target oriented, the target could function as the trainer, hence, RBS is used in this thesis for network synchronization.

## 6.4 Ultrasonic Sensor Scheduling

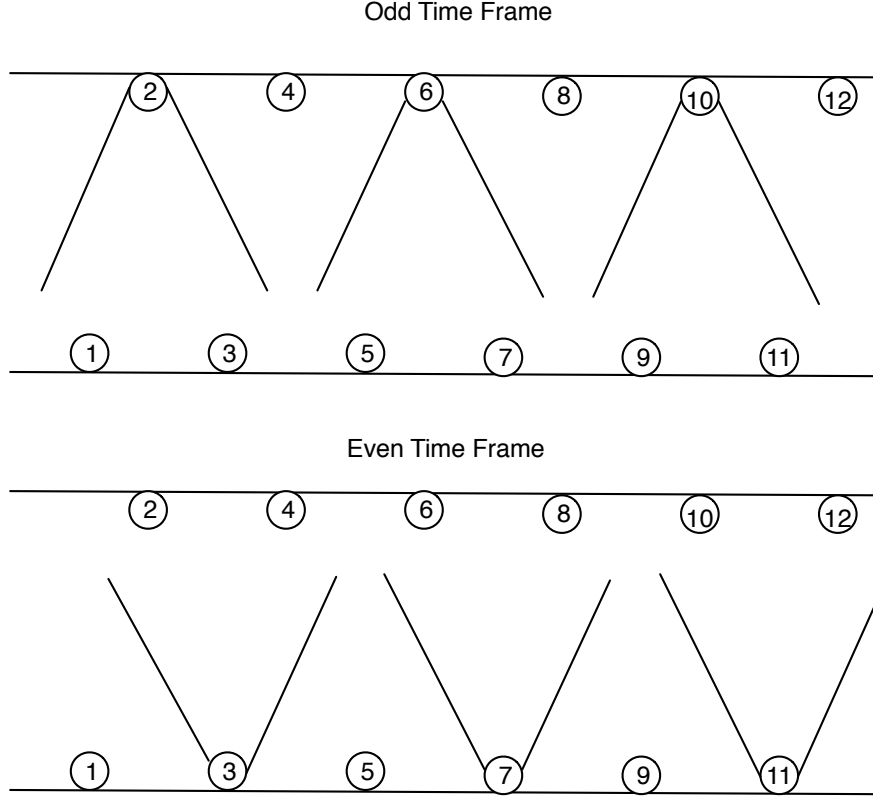
Using multiple randomly distributed ultrasonic sensors for target tracking is a difficult task. The ultrasonic sensors are directional instead of omni-directional, hence it is difficult to enforce a uniformed coverage region. Intelligently schedule the sensors will assign the sensors to monitor as large as possible with as little active sensors as possible. As mentioned in the self-localization section, there are two types of network discussed in this thesis, they are the network in the hallway, and in the rooms. In the hallway setup, the sensors are deployed in a more regular manner on the two boundaries. In the rooms, the sensors are mounted on the ceiling with the sonar facing down, the sensors are randomly deployed, and is triangulated using the method described in the self-localization section, then scheduling is performed on this triangulated graph. In this section, the scheduling of these two types of network is discussed, the scheduling schemes without the presence of a target are discussed first, and then the schemes with targets present are discussed.

### 6.4.1 Sensor Scheduling Without Target Present

When there are no targets in the sensor field, the sensors should primarily stay in sleep mode and periodically wake up to scan their sensing region for emerging targets, hence sensor energy can be conserved.

In the hallways situation, switching on all sensors simultaneously is not only energy inefficient, but also pose an interference problem due to the sensor region overlapping. In this situation, the sensors are switched on according to an alternative pattern as show in fig. 6.3. Only the sensors in the same side are switched on to avoid direct interference from the opposite wall. The sensors on both walls are used alternatively to avoid detection holes. The time frame size need to be selected so that it is long enough for the echoes to settle down, but short enough to avoid miss detection of a target. The effect of this value is studied in the implementation and experiment section.

In the room situation, where the sensors are mounted on the ceiling, hence there will be no direct interference between sensors, however, sensors with overlapping sensing region will receive the reflected signal. To avoid this over-hearing problem, only selected sensors are switched on at each time frame. After the triangulation in the self-localization phase, the three sensors in each triangle would have mutual overlap sensing regions, hence only one sensor from each triangle is needed to be switched on, and all its connected neighbors will stay off. The sensors that are switched on is referred as *active sensor* in the follow part of



**Figure 6.3:** Alternative switch pattern of hallway sensors

the thesis. The scheduling scheme for this situation is described in Algorithm 1.

In the algorithm, a random wait time is generated based the sensor energy level and the time it is not active, hence the sensors with low energy level are not likely to volunteer as an active sensor. The sensors that have already served active sensors will not volunteer to be an active sensor again, this will make the detection region more uniform over time.

### 6.4.2 Sensor Scheduling With Target Present

When target present, a target-oriented scheduling approach can be used for better sensor management. Assume the target is located in a triangle  $\alpha$ , in the next time frame, the sensors that are adjacent to the triangle  $\alpha$  are set to be active sensors, as shown in fig. 6.4 and fig. 6.5.

This target oriented scheduling scheme is only applied when there are target present in the

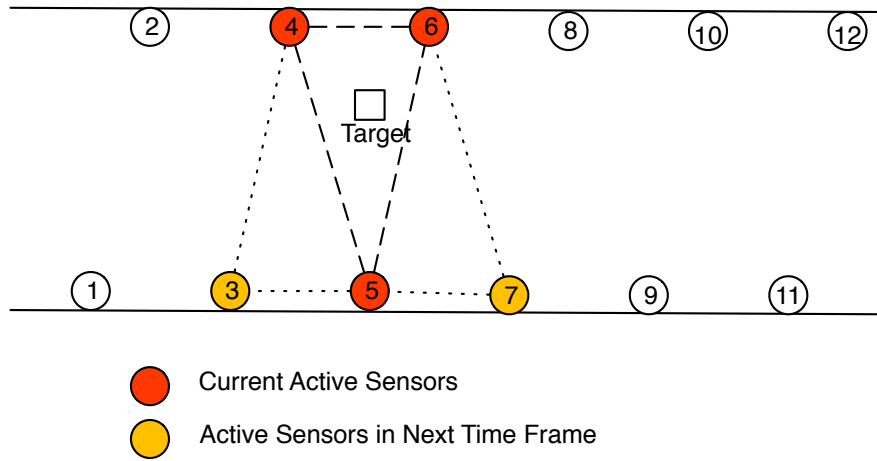
---

**Algorithm 3** Selection of active sensor in random ultrasonic sensor network

---

```
WaitTime  $\leftarrow$  random(My_Energy_Level + Last_Turn_Active)
while WaitTime  $\geq$  0 do
    Volunteer  $\leftarrow$  From_Radio
    WaitTime  $\leftarrow$  WaitTime - 1
end while
if !Volunteer then
    Volunteer  $\leftarrow$  MyID
    BroadcastVolunteertoneighbors
    Last_Turn_Active  $\leftarrow$  0
else
    Last_Turn  $\leftarrow$  Last_Turn_Active + 1
end if
```

---



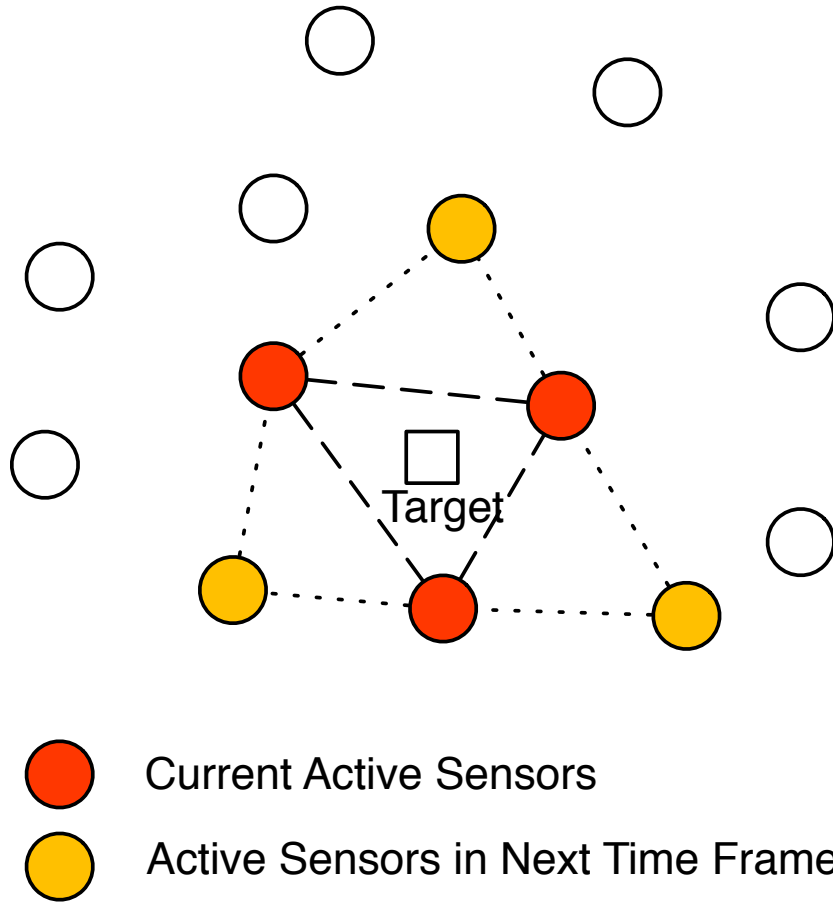
**Figure 6.4:** Target oriented sensor switch pattern in hallway

neighborhood, and the discovery of new target is broadcasted only to the neighbors, and the message will not broadcast to the entire network. The unrelated sensors in the network keep their routine periodic scan for emerging target using the scheduling scheme without target present. hence, emerging of new targets will be detected in a multi-target environment.

### 6.4.3 Target Localization Within Triangles

The target will need to be localized after it is detected by the active sensor. However, individual ultrasonic sensor can only obtain a distance measurement to the target. In a 2D situation, at least 3 distance measurements are required to estimate the location of the target.



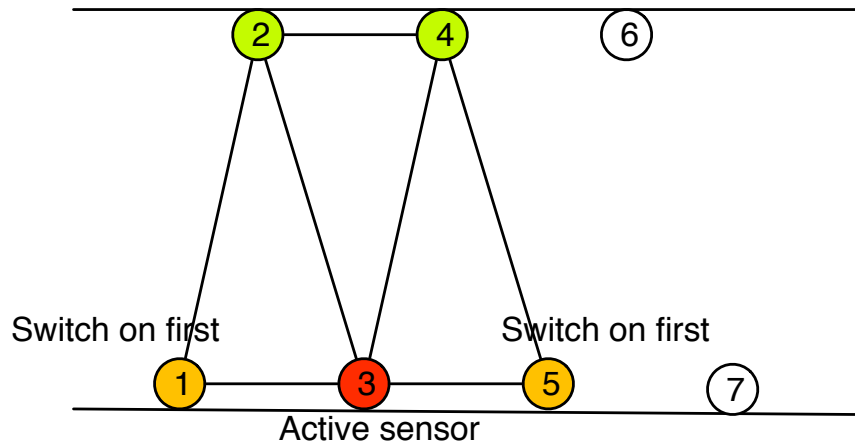


**Figure 6.5:** Target oriented sensor switch pattern in randomly distributed network topology

Hence, sensors that are near the target need to be waken up to take the measurements. Ideally, we could only wake up the two sensors that are forming a triangle with the active sensor, and the target is located within the triangle. However, the active sensor maybe belongs to multiple triangles, and the target could be in any one of these triangles. It is not feasible to test through these triangles one by one, because in this neighborhood, only 1 sensor should be switched on at a time to avoid interference, search through all these sensors will take a long time, and the target would have moved away from the location.

In hallway setup, there are three possible triangles around an active sensor as shown in the early figures. The situation is relatively easy in this case, we can switch on sensors based on the pattern showed in figure 6.6. Suppose sensor 3 is the active sensor, sensor 1 and 5 can be switched on simultaneously to check if the target is within their vicinity. If one of them detects the target, then we can decide which triangle the target is in. If neither of

the two sensors detects the target, then we can decide that the target is within the middle triangle, hence sensor 2 and 4 will be switched on respectively.

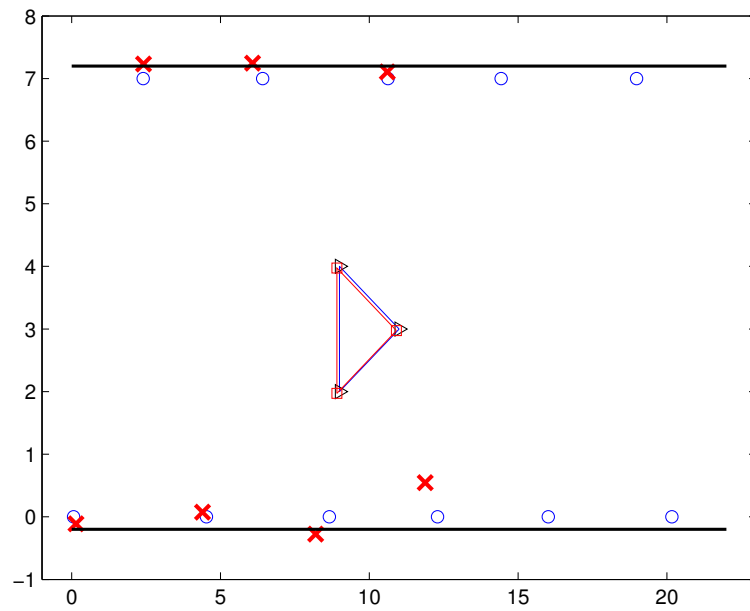


**Figure 6.6:** Sensor switching pattern in hallway after the active sensor is identified

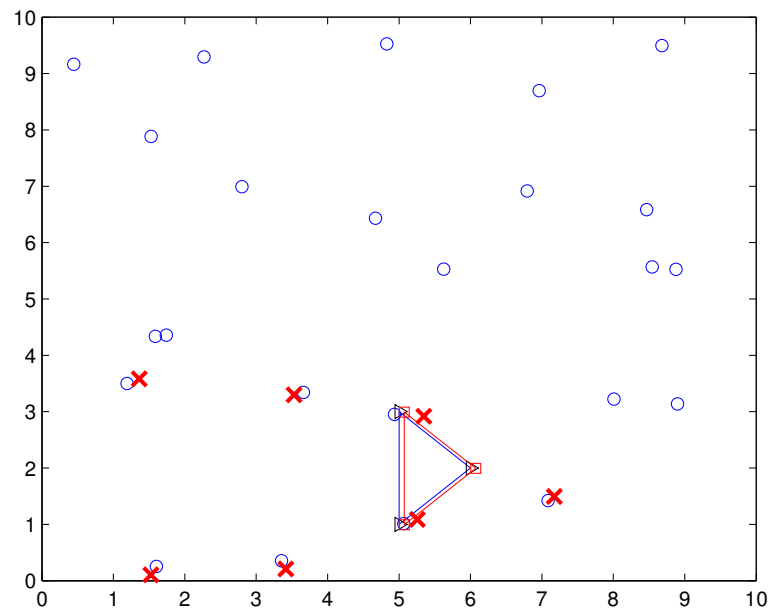
## 6.5 Simulation Results

The self-localization scheme is simulated in two scenarios. In the first scenario, the trainer has a smaller triangle of side length 2 ft, while the second one has a larger triangle of side length 4 ft. The target travels through the network with a constant speed of 2 ft/s, and this speed is measured in every time frame. A  $\pm 5\%$  measurement noise is added to the speed measurement. The trainer's position is calculated using this measured speed, and the each sensor is able to measure its distant to each vertex of the trainer, these distances are used to localize the sensor itself. The measured distances have zero mean Gaussian noises added. Fig. 6.7 and Fig. 6.8 are snapshots of the situation in process, where the blue triangle is the trainer, the red triangle is the calculated trainer position, the circles are the actual sensor location, and the "x"s are the estimated location of the target. In this simulation, the variance of the Gaussian noise is 0.1.

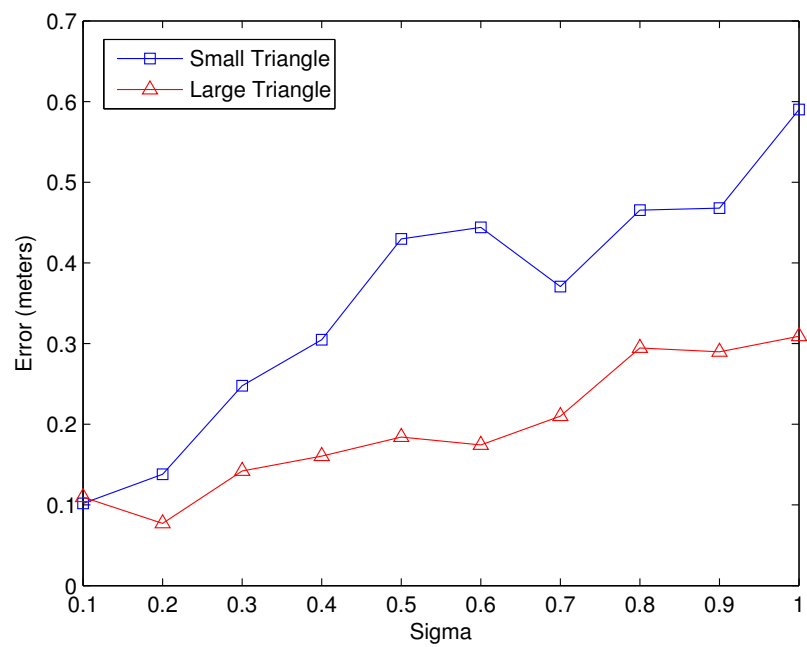
We vary the variance of the Gaussian noise from 0.1 to 1, and the localization error are computed using MMSE. Fig. 6.9 shows the estimation error against different  $\sigma$  value. The estimation error for each  $\sigma$  is averaged between 50 runs. The large triangles generate more accurate locations.



**Figure 6.7:** Snapshot of self-localization in hallway



**Figure 6.8:** Snapshot of self-localization in an open space



**Figure 6.9:** Estimation error for sensor locations

## **Chapter 7**

# **Falling Detection in Elderly Homes with Radar Sensor Networks**

### **7.1 Introduction**

As an aging society, the necessity of intensive care for elderly individuals have been grown faster than ever, the current service infrastructure are severely challenged. It is desired to develop technologies that can tender the need of the elders automatically with minimal human supervision, hence the elders can remain at home for extended period of time, reducing the crowds in the hospital and nursing homes. The system would also provide the convenience and privacy of homes, hence offering the elders a completely new lifestyle. However, there are several hazards that are associated with this independent lifestyle of the elderly people, among them, the falling incident is one of major causes of server injuries and death of the elderly people that are living independently. It is estimated that there are over 30 percent of falling incident happens people over age 75 each year [92]. It is desirable to have a system that will report the real-time status of the elderly people, and automatically call for assistance when falling incidents occur. There are several existing systems that allow elders to call for assistance by pushing a button on a wearable device in emergency, however, in many cases, the falling will result in unconsciousness, hence the elder unable to push the button. Other pre-existing conditions such as dementia also prevent elders to be able to operate the device. A falling detection system is desired, which can efficiently determine when falling occurs and when assistance is needed.

There are generally two types of systems that are commonly used in falling detection. The one type is the wearable sensors, and the other type is the non-wearable sensor network

based system. Both of them have advantages and limitations. Wearable falling detectors are usually referred as body area sensor networks, where a number of sensors are mounted on the several key parts of the body. Non-wearable fall detectors usually have the sensors integrated into the house, and collect elder's postures through video and sound.

For wearable sensor network based systems, accelerometer is usually the sensor choice. Some system monitors the differences of instantaneous acceleration from the hand, waist and ankle to predict a falling process [93] [94], and some system simply detect the direction of gravity from the three accelerometer, and match these readings to known body postures, such as lying, sitting and standing, and use HMM to determine whether the subject is falling or simply lying down [95] [96] [97] [98]. These systems usually need to mount bulky parts on several different body parts, hence making it very inconvenient for the elderly people to wear. Newer techniques integrated the hardware platform to a single sensor, usually a watch-sized device that contains a 3-axis accelerometer and inertia sensors [99] [90], some system even directly utilized cellphone to perform such function, since many smart-phones already integrated the accelerometer on them, with cellphones, the collected data can be send out directly using the cellular network [100] [101]. Supporting vector machine technique is usually used to classify the activity of the subject that wearing the system. In these systems, the user need to wear only a single device, which is a better alternative, but since the subjects are the elderly people, it is very likely that they will often forget to wear the device or turn the device on.

As for non-wearable sensor network based system, smart camera networks are usually used [102] [103]. In these systems, two tasks are typically performed, first is to determine whether there is a subject within the camera's view, if the subject is presenting, the system then determines the posture of the target. Presents of the subject is usually done with background subtraction, then the key body parts (i.e. head, foot, trunk) are identified, and their relative positions are computed according to the axis system defined by the camera, and the onboard processor will determine the posture based on these relative position [101] [104] [105]. These techniques are essentially centralized method, where discrete images from a single camera are used to make the inference. The networked camera is just used to extend the coverage region instead of providing collective conclusion. The major drawback of using single camera to make inference is the occlusion problem, because single camera has no way of seeing what is behind the obstacles. More advanced system utilizes the computer vision techniques to determine the body posture. They model the human body with a graphical model after identifying key body parts, each part in the body is modeled by a node [106]. The precise body movement is determine through the transition of the nodes, hence a more comprehensive view of the body posture can be obtained, and the future postures can be predicted using Bayesian estimation. In these systems, even some body parts are occluded in some instances, the entire body posture can still be estimated, and the occluded part can be estimated using historical data.

With the advancement of 3D graphics technology, some systems start to reconstruct 3D scenes from 2D images to predict human postures [107] [108]. Once the 3D model of the people is reconstructed, the posture of the subject is known. In order to perform this technique, multiple cameras must have overlapped views from several different angles, and the onboard processor must be powerful enough to perform the reconstruction, since the computation is quite intensive. These systems are quite complex, and are usually used for greater uses than simply falling detection. Some other systems for none-wearing sensors utilize different types of sensors rather than cameras for falling detection, one of them is using audio for falling detection [109] [93]. This system trains the sensor node with different sound generated by the subject with doing different things, such as walking, clapping hands, sitting, and falling. When the subject is presenting, the sensor will capture the sound generated by the subject, and apply a match filter to determine what the subject is doing. This system is not suitable for noise areas, since the sound generated by the subject can be easily corrupted.

In this chapter, a bumblebee radar network based falling detection system is presented. The system is able to detect falling people using location and speed measurement obtained through a small embedded radar system. The network are connected through base-station node that will forward the timely data to the computer, then transmitted to the appropriate personnel. The hardware platform presented in this chapter is an array of bumblebee radar network for speed and location detection. The system itself is integrated to the house infrastructure, hence no wearable device or cooperation of the elderly is required. Differ from the camera network based system, the radar system is less dependent on the illumination of the room, and is less likely been occluded by obstacles of different shapes. The camera needs to be able to see a substantial part of the elderly people in order to decide the status of him/her, but the radar system need only to measure the speed profile of the body parts, even if the body of the elder is partially occluded, the radar system is still able to function properly.

## **7.2 System Architecture**

### **7.2.1 Automated falling detection system**

A probabilistic falling detection system should be able to compute the likelihood of elderly falling events using the location information and the body motion of the elderly people. Although there are many ways to obtain the position information and body movements of the elderly people, this automated system should be produce as little disturbance to

the elderly as possible, should not require the elderly people to cooperate, and should not require modification for the existing infrastructure of the elderly homes. With such requirement, a network of radar sensors is used for this purpose. Each of the radar has limited sensing regions, thus providing the rough location of the elderly people, and it is able to capture the body speed of the elderly people.

In this thesis, we propose a two layer system as shown in Fig. 7.1. The bottom layer of the system consists of individual bumblebee radar with an embedded microprocessor, the raw data from the bumblebee radar is processed by the microprocessor to make a detection decision locally and independently. The upper layer will be gateway computer which act as a fusion center. The fusion center will fuse the location of the sensors with the individual decision from the lower layer. This is because the radar can only detect movement in the radial direction, the target may be “invisible” to some radar modules due to its movement direction. Hence the overall falling decision should be made collectively by all nearby sensors. In the local detection layer, a Hidden Markov Model is constructed, and Viterbi algorithm is used to make inference about possible states of the elders. The data fusion layer is a weighted majority vote system, where a cost function is computed based on the detection probability of the nearby sensors.

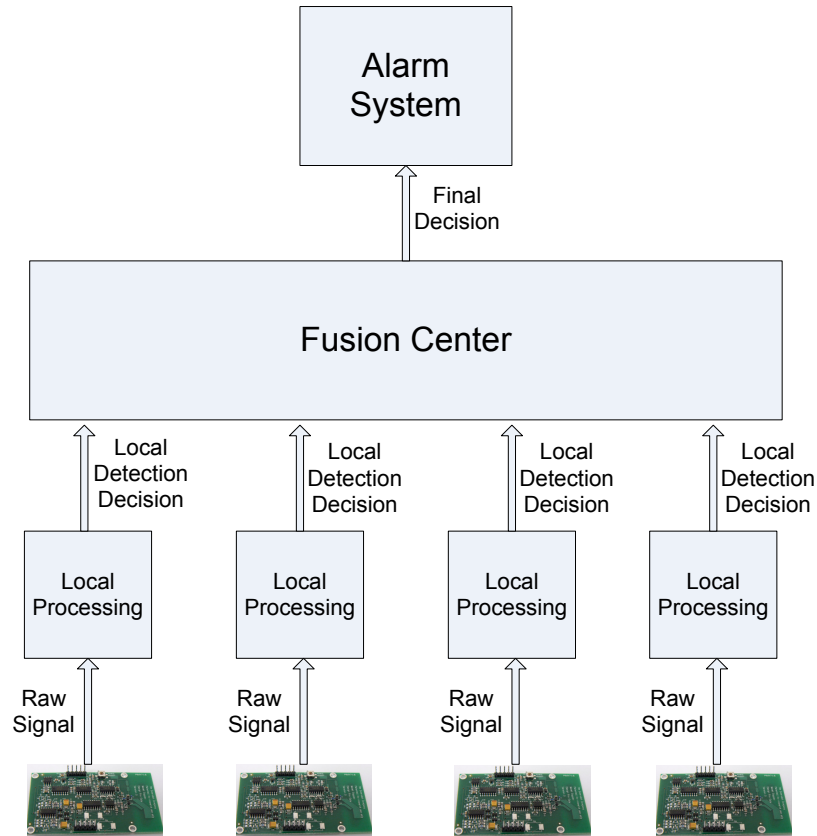
The rest of the chapter is organized as following, we first discuss the hardware platform that is used to construct this radar sensor network framework in this section, the hardware architecture of the sensor board, and the deployment pattern of the sensor with-in an typical apartment setting. In section III, we formulate local detection system by develop the Hidden Markov Model, and identify the possible states of the elderly people. In section IV, we use the Viterbi algorithm to infer the falling states of the elderly people on the hidden Markov model constructed. In section V, the functionality of the fusion center will be discussed. In section VI will show some statistics collected from mocking falling scenario.

## **7.2.2 Hardware Platform**

### **7.2.2.1 Radar Sensor Board**

Bumblebee radar sensor is used in this chapter, the sensor itself is small size, low power, and can be easily integrated with processing and wireless transceiver system such as telosb sensor mote, this will allow easy construction of a ad hoc mesh network of the sensors. The bumblebee radar sensor itself has a maximum measurement range of 10 meters, and it has a conical measurement region of 60 degrees. The radar operates at a center frequency of 5.8 GHz, and is be able to detection radial motion velocity of as small as 2.6 cm/s. Fig 7.2

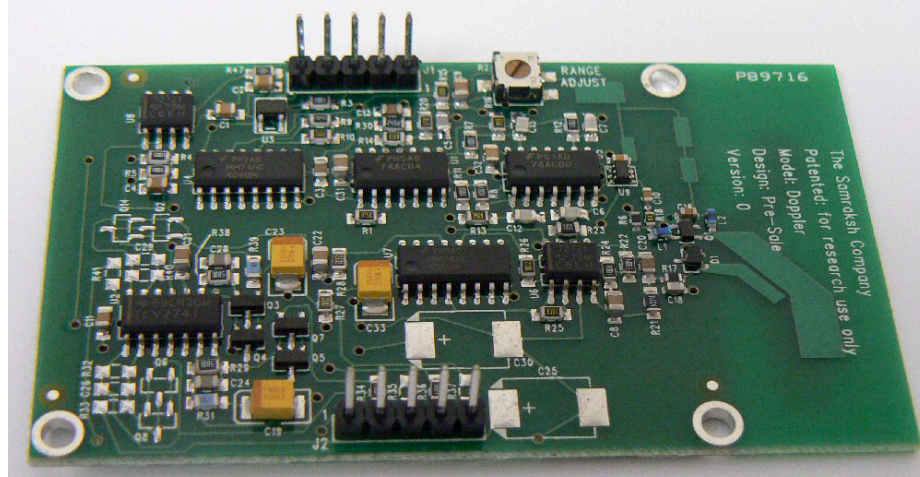




**Figure 7.1:** System Architecture

show the sensor board itself, it measures less than 10 cm diagonally, which is very small for typical wall mount, and will not interfere with any of the normal living activities inside the house.

In this chapter, we selected TI EZ430 sensor node to interface with the bumblebee radar to act as a data processor, and wireless transceiver. The EZ430 is equipped with a MSP430-r2274 CPU with two analog input ports for interface with the bumblebee radar. The EZ430 kit is equipped with CC2500 radio chip for wireless communication, which is fully compatible with 802.15.4 protocol. Fig 7.3 shows the EZ430 board. The EZ430 is powered by 2 AAA batteries, and the bumblebee radar is powered by 4 AAA batteries, so each individual sensor node is self-contained without the needs for external power source.



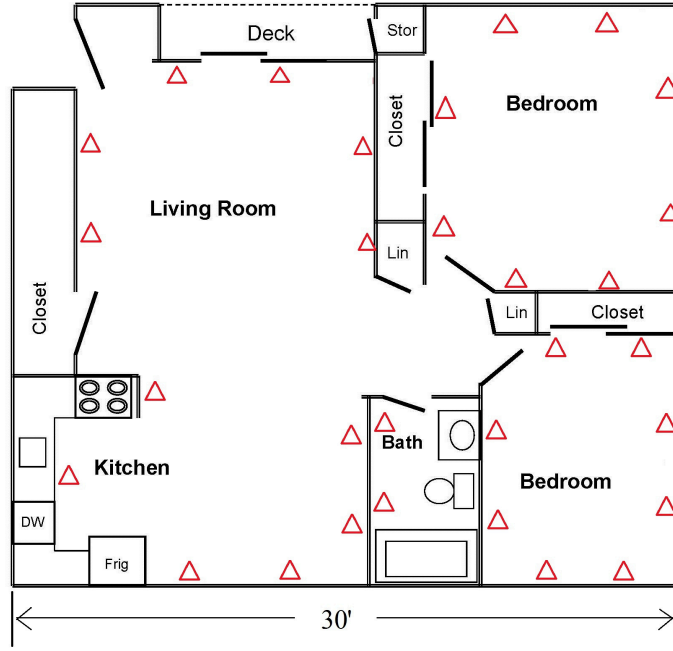
**Figure 7.2:** Bumblebee radar used in this chapter



**Figure 7.3:** MSP 430 sensor board used in this chapter

### 7.2.2.2 Sensor Deployment

Since the bumblebee radar has a 60 degree cone-shaped detection region, we would like to arrange the sensors to be 4 feet apart, so that the sensors will have some overlapped about 4 feet away from the wall. This will ensure that the sensing regions of the radars are continuous throughout the house, and it could provide some redundancy for possible detection failures. Fig 7.4 shows a typical 800 square foot two bedroom apartment, and a total of 30 sensors are implemented in the apartment.



**Figure 7.4:** Possible state of the elder's activity

## 7.3 Local Detection

In this section, the local detection algorithm is developed by first examine the raw data produce from the bumblebee radar sensor, and then propose several states of the elderly activity. The data and the states are then formulated into a hidden Markov model. With the hidden Markov model, each bumblebee radar will act as an individual detector.

### 7.3.1 Bumblebee Sensor Model

Bumblebee radar is a small scale pulse Doppler radar, which reports the radial speed of the target based on the phase shifted Doppler frequency.

$$f_d = 2 \frac{v_r}{\lambda}, \quad (7.1)$$

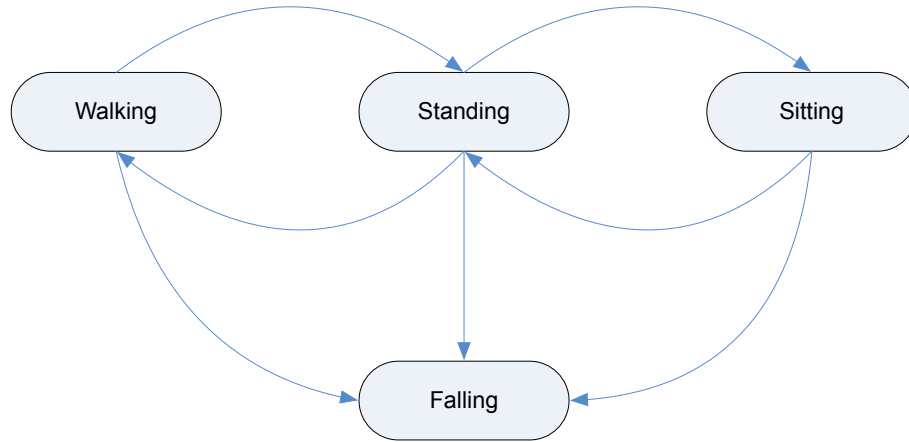
where  $f_d$  is the Doppler frequency shift,  $v_r$  is the speed of the target, and  $\lambda$  is the wavelength, which is depended on the operation frequency of the Bumblebee radar.

The operation frequency of the bumblebee radar is  $5.86G$ , hence the speed of the target can be determined as  $v_r = f_d/39$ . The Doppler frequency shift  $f_d$  is measured by the bumblebee radar, and transmitted to the EZ430 board via two analog inputs. The measured Doppler frequency shift is reported in terms of demodulated signals  $I$  and  $Q$  values. Cross-correlation method is used to parse this demodulated signal locally on the sensor board.

## 7.4 Elderly Activity States

The bumblebee sensor is mounted at the height of the waist level, so that it is observing mainly on the trunk of human body, hence avoid detecting most of the random gestures by the limbs while the elderly person is stationary. With such a system, the radar sensor itself can only make inference based on simple speed profile of the body trunk.

By analyzing the speed profile of the elderly people, we propose four general states for the elderly people staying in the house, they are *Standing*, *Walking*, *Sitting*, and *Falling*. The elderly people may stay in one of the states or may transit from one state to another state with certain probability, as shown in Fig 7.5. The local detection algorithm will focus on how to determine the elderly activity state transits into the falling state.

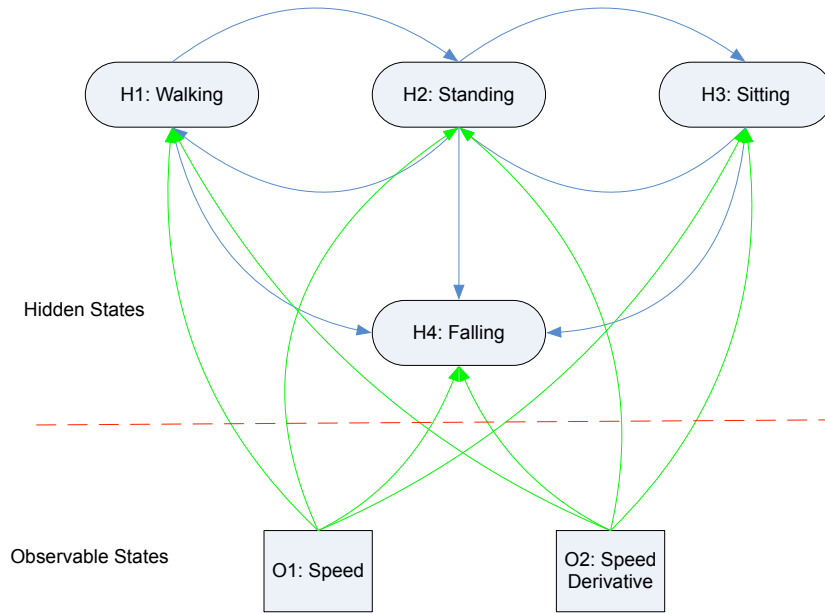


**Figure 7.5:** Possible state of the elder's activity

### 7.4.1 Hidden Markov Model

The hardware itself can only provide speed measurement, it is not capable to directly tell which state the elderly is in. In order to infer the activity state, a hidden Markov model is constructed.

A hidden Markov model contains two sets of states, the hidden states and the observable states. The hidden states in this situation would be the activity states the elders are in, hence the *Standing*, *Walking*, *Sitting*, and *Falling* states are all hidden states, these states are connected according to Fig 7.5. The observable states are the information we collected from the sensor network, which are *location*, *displacement*, *instantaneous speed*, and *speed change*. These states are connected to the hidden state through observation distributions, as shown in Fig 7.6.



**Figure 7.6:** Hidden Markov model for elder's activity

Let the hidden Markov model be characterized by parameter set:

$$\lambda_t = \{A, B, \delta_t\} \quad (7.2)$$

where  $A$  is the state transition matrix of the hidden states, and  $B$  is the observation matrix, and  $\delta$  is the state vector. Since we have four hidden states, matrix  $A$  is a 4 by 4 matrix containing all the probabilities to transit between  $H_1$ ,  $H_2$ ,  $H_3$ , and  $H_4$ . As one may note, the arrow pointing from  $H_4$  to  $H_1$ ,  $H_2$ , and  $H_3$  are missing, this is due to the assumption that once the elder falls down, he will remain in the falling state, as the reflection in the  $A$  matrix,  $A(4,1)$ ,  $A(4,2)$ ,  $A(4,3)$  are all set to 0 all time. Since there are three observation states, matrix  $B$  is a 4x2 matrix, containing all the probabilities that links between the  $O$  states and the  $H$  states.  $\delta$  is a 1x4 vector that indicate which state the elderly people is most likely be in. The  $t$  subscript denotes the time frame, at the beginning, the elderly people is assumed to be in a known state denoted by  $\delta_0$ , and the HMM is denoted at  $\lambda_0$ .

### 7.4.2 Parameter Assumptions

As described by the Hidden Markov model, the state probability  $\delta_t$  indicates the most probable state. In order to determine  $\delta_t$  for each time frame, matrix  $A$  and  $B$  must be defined. Note that matrix  $A$  and  $B$  are static once they are defined, they will not change with time.

Matrix  $B$  is the observation matrix modeling the probability linking the observation states and the hidden states. Each entry in the matrix can be represented by a probability expression  $P(H|O)$ .  $O_1$  measures the speed of the elderly people, if the speed is fairly large, the elder is either moving or falling, and is not stationary, so the probability is equally divided between state  $H_2$  walking and state  $H_4$  falling. In here, speed is used as a binary entry, where  $O_1 > \tau_s$  is used to produce either a 1 or a 0. This is to filter out the small movement of the body to be considered as a moving action.

$$P(H_1|O_1) = \begin{cases} 0.8 & \text{if } O_1 > \tau_s; \\ 0.2 & \text{otherwise.} \end{cases} \quad (7.3)$$

$$P(H_2|O_1) = \begin{cases} 0.8 & \text{if } O_1 < \tau_s; \\ 0.2 & \text{otherwise} \end{cases} \quad (7.4)$$

$$P(H_3|O_1) = \begin{cases} 0.8 & \text{if } O_1 < \tau_s; \\ 0.2 & \text{otherwise} \end{cases} \quad (7.5)$$

$$P(H_4|O_1) = \begin{cases} 0.8 & \text{if } O_1 > \tau_s; \\ 0.2 & \text{otherwise} \end{cases} \quad (7.6)$$

The observable state  $O_2$  is speed derivative. It measures the change in speed of the elderly people, is the essential factor to differentiate walking state and falling state. A typical factor would look like

$$P(H_1|O_2) = \begin{cases} 0.8 & \text{if } O_2 < \tau_d; \\ 0.2 & \text{otherwise.} \end{cases} \quad (7.7)$$

$$P(H_2|O_2) = \begin{cases} 0.8 & \text{if } O_2 < \tau_d; \\ 0.2 & \text{otherwise} \end{cases} \quad (7.8)$$

$$P(H_3|O_2) = \begin{cases} 0.8 & \text{if } O_2 > \tau_s; \\ 0.2 & \text{otherwise} \end{cases} \quad (7.9)$$

$$P(H_4|O_2) = \begin{cases} 0.8 & \text{if } O_2 > \tau_s; \\ 0.2 & \text{otherwise} \end{cases} \quad (7.10)$$

$\tau_d$  is the threshold for speed derivative.

Matrix A is the transition probability between states, this matrix can be obtained by supervised learning process based on the living habits and behavior of the subject. Intuitively the elderly people is more likely to fall while walking than standing still, hence, if we assume that the probability for the target to transit from walking to falling is 0.4, and from standing and stationary to falling is 0.1, and the probability are equally divided between all other transitions, the Matrix A would look like this,

$$\begin{aligned} P(H_4|H_1) &= 0.4 \\ P(H_3|H_1) &= 0 \\ P(H_2|H_1) &= 0.3 \\ P(H_1|H_1) &= 0.3 \end{aligned} \quad (7.11)$$

The probability distribution for other hypothesis can be assigned follow similar patterns.

## 7.5 Fall Detection Using HMM and Viterbi Algorithm

Once the Hidden Markov model is obtained, the Viterbi algorithm can be applied to determine the most probable state for each time frame, this is done by computing the new  $\delta_t$  based on  $\delta_{t-1}$ . For each state  $i$ ,  $\delta_t(i)$  indicate the probability of the elder being in state  $i$ . To compute  $\delta_t$  for each state, in time frame 1 ( $t = 1$ ),

$$\delta_1(i) = \delta_0(i)B(i, k_1) \quad (7.12)$$



where  $k_1$  is the corresponding data reading obtained from the two observable states at time 1. For  $t = 2, \dots, T$ ,

$$\delta_t(i) = \max_j (\delta_{t-1}(j)A(j, i)B(i, k_t)), \quad (7.13)$$

where  $j \in \{H1, H2, H3\}$ .

The current state of the elder can be computed using

$$i_t = \arg \max_i \delta_t(i) \quad (7.14)$$

at each time frame.

Since the system halts and report problems once it detects a falling event, it rarely need backtracking phase of the Viterbi algorithm, but backtracking phase should still be implemented in order to recover any possible miss detection of falling event. The backtracking process is to estimate the previous state (i.e. state at  $t-1$ ) given current state.

$$j_{t-1} = \arg \max_j \delta_{t-1}(j)A(j, i) \quad (7.15)$$

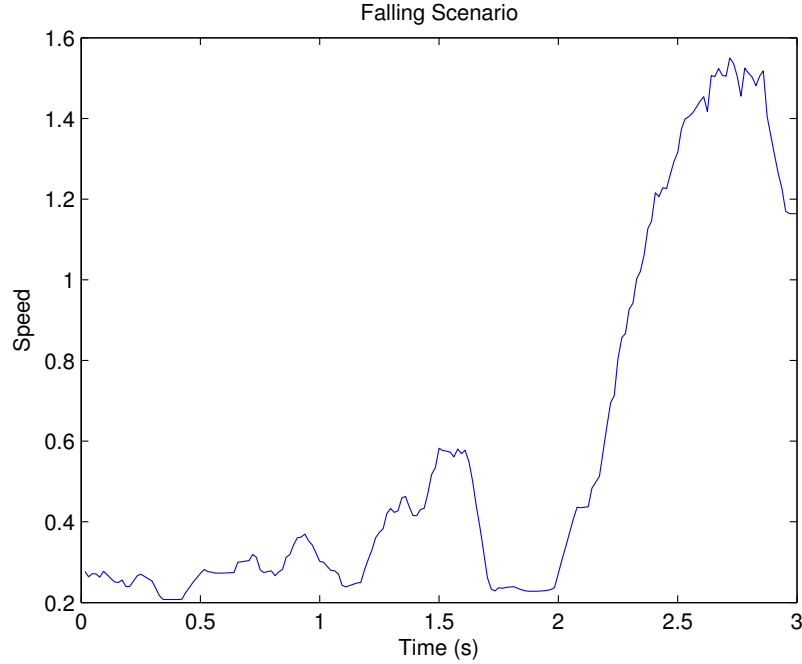
$j$  can be compared with the previously determined  $j$  to decide whether a miss detection has occurred. This process is repeated from  $j_{t-1}$ ,  $j_{t-2}$  to  $j_1$ , to verify all past data.

## 7.6 Experimenting with Local Detection Decision

To verify the local detection decision we obtained, a few speed profiles measured using bumblebee radar is studied and presented.

The first scenario is a typical falling scenario captured by the bumblebee radar, the author stand in front of the radar, and performed a falling action. The speed profile can be is shown in Fig. 7.7.

In this profile, the state vector probability is shown in Fig. 7.8, initially as the falling start, the dramatically increase of the speed indicated either a falling action or a walking action, and as time goes on, the algorithm eventually decide that it is more like a falling action than walking. As you can see, the probability for walking and falling increase and decrease over time.



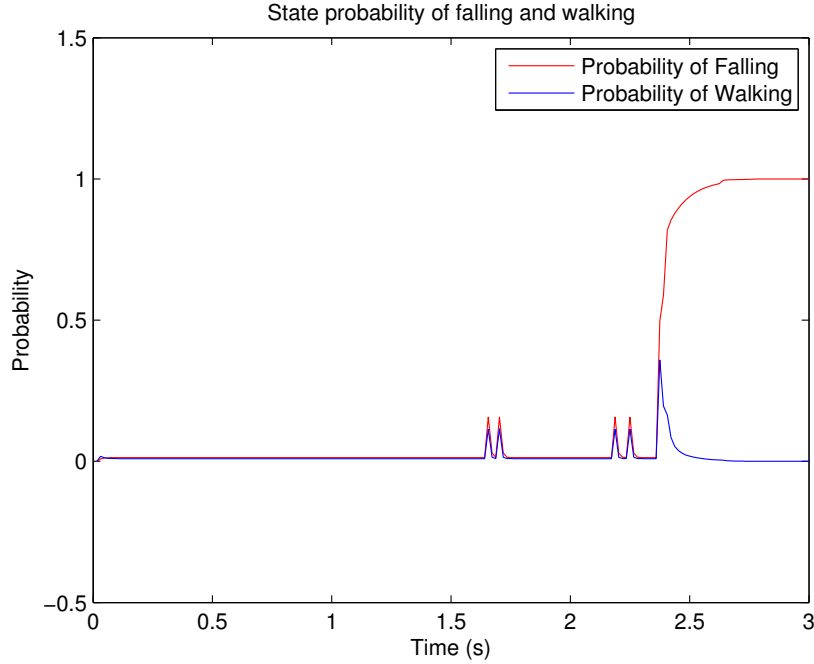
**Figure 7.7:** Speed profile for a falling scenario

The second scenario is actually a walking to standing transition, the speed profile is shown in Fig. 7.9.

For this profile, the state vector probability is shown in Fig. 7.10. In this scenario, the initial state was wrongly set at standing state, and the algorithm itself determined that it is actually in a walking scenario.

## 7.7 Fusion Center

Although individual sensors are capable of detecting falling state of the elderly people, however, there can be scenarios which are difficult for individual sensors to distinguish between a falling state and walking state. For instance, in our proposed 2D plane, radar sensor can only detect the motion that is in the radial direction of its facing direction, if the subject is walking toward the radar initially, and then changed his direction of walking at some moment, the radar sensor would detect a sudden drop in subject's movement speed, and the individual radar would tend to believe it is an falling action. A fusion center, which combines one or more detection result from nearby radars would make corrections to individual sensor decisions to produce a collective inference.

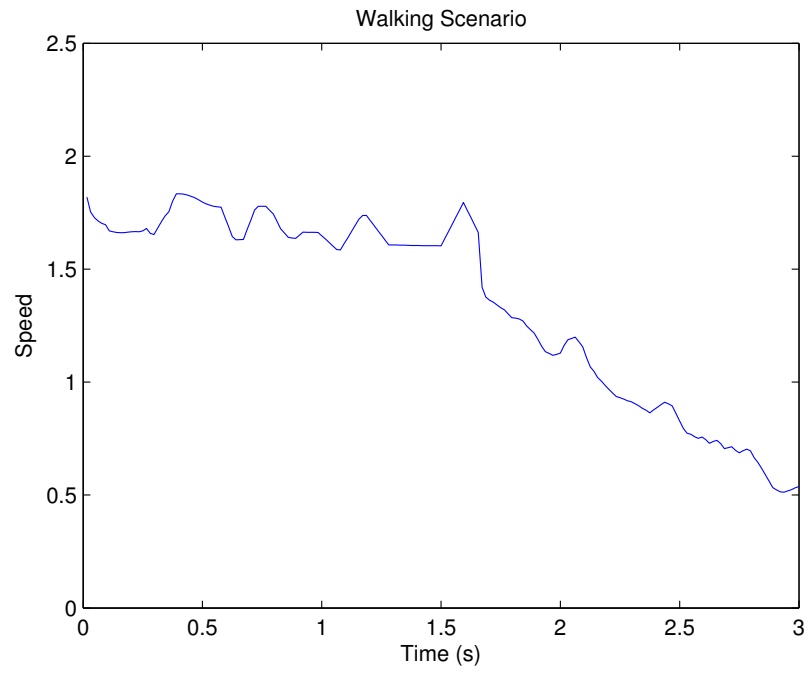


**Figure 7.8:** State probability for falling and walking

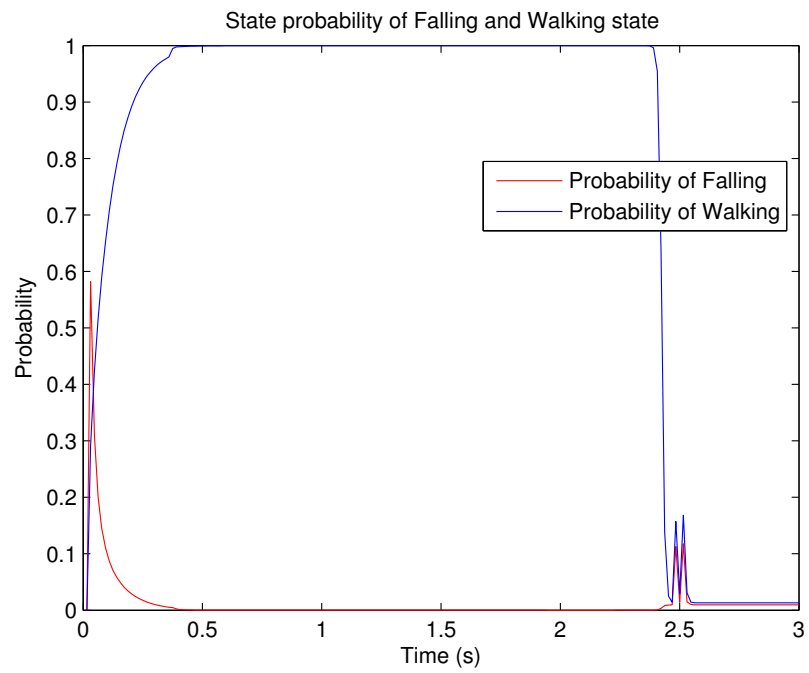
Since the sensors are deployed in the room with prior knowledge about sensor location and their facing direction, we can easily group the sensors into clusters in the fusion center. In here, we consider two types of sensor arrangement. In the first type of arrangement, two bumblebee radars are arranged to facing each other. In the second type of sensor arrangement, the two radars are arranged with a 90 degree angle.

As described in the previous paragraph, in the fusion center, two extra parameters are included in addition to the local detection decision, they are sensor location and sensor facing angle. If the sensors are facing each other, then their decision should be redundant to each other, hence we need to perform a *intersection* operation to the two decision data. If the sensors are not facing each other, their data should be complimentary to each other, hence an *Union* operation is needed to incorporate data from both sensors. An illustration of the fusion center is illustrated in Fig. 7.11.

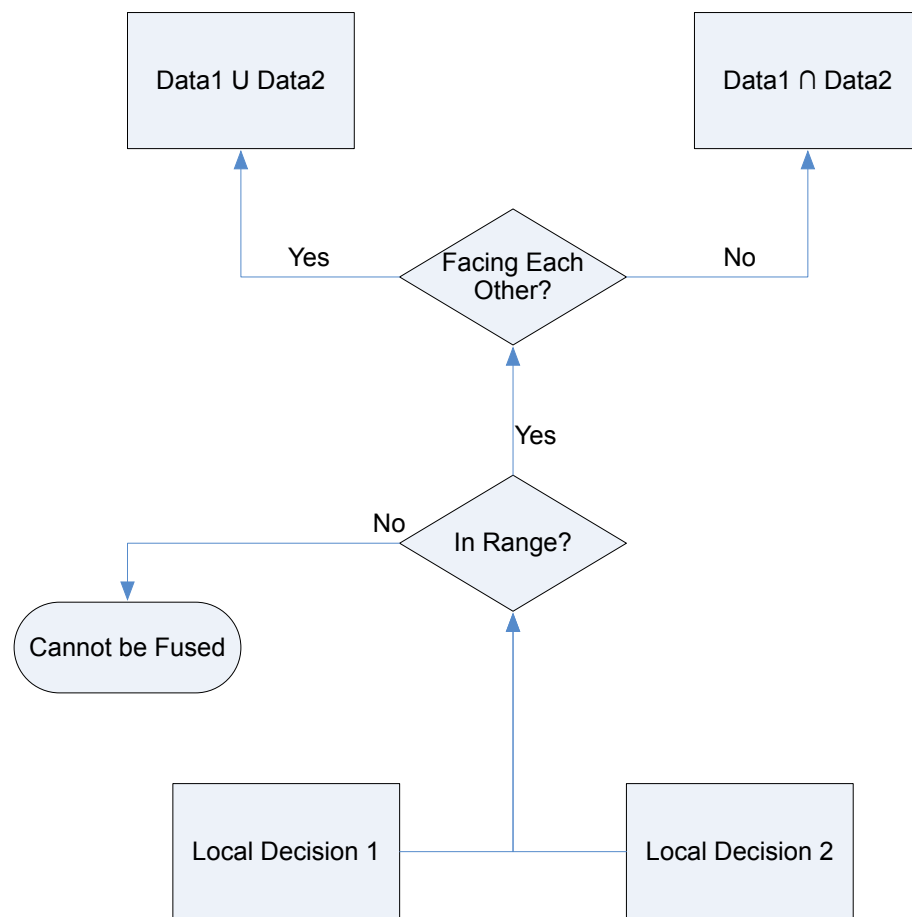
As for hardware, a gateway computer is used as the fusion center, and the data are stream into the gateway computer through wireless connections. The sensors will broadcast periodic signals to the gateway in a TDMA fashion. Each sensor is expected to have only a small number of messages to transmit to the gateway, however, if the gateway has decided that a falling event has occurred at a sensor location, it would grant the sensors in that area exclusive rights to transmit uses a semaphore-like mechanism.



**Figure 7.9:** Speed profile for a walking to standing scenario



**Figure 7.10:** State probability for falling and walking



**Figure 7.11:** Functionality flow chart of the fusion center



# Conclusion

This thesis focuses on a classic application of wireless sensor networks, multiple-target tracking. A fast, low cost target tracking framework is developed specifically for wireless sensor networks. It takes all the physical limitation of the wireless sensor network into consideration, and it provides high accurate tracking result while consuming less energy. The thesis starts out from the two-tier target tracking algorithm on a simple chained formed network, and expends the framework out to the 2-D, full randomly distributed network topology. It takes advantage of the redundancy in the wireless sensor network, and it utilizes the graph model concept and the conditional independency in Markov random field to manipulate the randomly distributed network to fit the two-tier framework. The manipulation techniques are not constrained by the dimension; hence the algorithms can be expanded to the 3-D world if necessary.

In addition to the development of the target tracking framework, this thesis also provides basic techniques for self localization and synchronization, so that the sensor network can be calibrated autonomously after random deployment. With these techniques, the sensor network will be ready to perform the target tracking functionality with minimum human interaction. The thesis also provides an example of a possible real world use case for the target tracking framework using an ultrasonic/low energy radar hybrid sensor network.

## Contribution

In Chapter 3 of this thesis, a new low-complexity two-tier target tracking framework for distributed sensor networks is developed. The framework is fast and low cost; it can be applied to different sensor topology under different space constraints. The framework utilizes the efficiency of the binary sensor readings to isolate the target in tier one, and then it uses a small cluster around the isolated target obtained in tier one to perform more accurate local estimation in tier two. Tier one algorithm represent a typical continuous tracking process, but with low cost and low accuracy binary sensor data. Tier two represents

a discrete localization process. By combining the two tiers into a single framework, we obtained the best of two worlds, the continuity and low cost of the binary tracking, and the accuracy of the localization. There are only one other work that utilized a similar two tier technique to perform target tracking, however the tier two algorithm they used is far less accurate. Our original work was published in IROS 2007, as the other work [58], is published in 2011.

Chapter 4 serves as a transition in this thesis. The main contribution of Chapter 4 is to expand the two-tier framework developed in Chapter 3 to a 2-D, randomly distributed sensor network. In this chapter, the concept of using graphical models and cliques to represent randomly distributed sensor network is conceived. This chapter links the two-tier framework in Chapter 3 into the next chapter, where the technique for manipulation of the network topology is developed.

Chapter 5 models the sensor network as a Markov random field (MRF), and then it converts the randomly distributed sensor field into a clique network with a grid-like topology. After the grid-like topology is setup, the two-tier framework is applied to track the targets. Since a nice grid-like framework is setup, for tier one algorithm, we introduce a simple search algorithm similar to the image processing search technique, which is very low cost, and the computation can be performed in a parallel fashion among the sensors. Since each node in the grid-like topology is a clique containing multiple sensors, they serve as a natural local cluster for localization, hence falling under tier two parts for the two-tier framework. The importance of the grid-like topology construction is much more than simply applying two-tier framework to it. It effectively isolated the network into conditionally independent clusters, hence opening a whole new range of possibilities for dynamic topology manipulation.

Despite the name “grid-like”, the contribution of this chapter is not only the “grid-like” part. The second contribution of Chapter 5 is that we partition the network into conditional independent cliques, and soften the space/distance correlation between the individual clique and the rest of the network; hence we can move the clique position in the network topology map without affect the physical location of the individual sensor. This opens the door for a wide range of opportunities in dynamic network topology manipulation.

Chapter 6 and Chapter 7 provide an example of possible implementation and application of the two-tier framework and grid-like network structure. Chapter 6 lays down the gravel for tracking; it shows how to localize the sensors in randomly distributed sensor network with little prior information. Chapter 7 shows how to use the localized sensor network to perform falling detection in a typical apartment setting.



## Results

The two-tier framework was evaluated in Chapter 3 on a chained-form network, where 100 sensors are arranged in a corridor setting, spaced at 1 meter away from each other. When target passes by, it is isolated into a 5-sensor cluster by tier-one algorithm, and then maximum likelihood is performed on the cluster to find the exact location of the target. After testing the framework on the chained-form network, we then re-arranged the sensors into a hexagon-shaped network, and grid-shaped network to demonstrate the two-tier framework can be adapted to various network topologies.

In these tests, we assumed Gaussian noise model with  $\sigma = 0.1$ , and the MMSE is computed for the each of the network topology. The estimation error varies from 0.06m to less than 0.1m, with the grid-shaped network being the least accurate to the chained-formed network to be most accurate. In comparison to [58], where the "triplet circle intersection" (TCI) algorithm is developed, the TCI algorithm assumes only binary sensor and perfect sensing range with no noise considered, we still achieve a much better location accuracy.

Chapter 4 expands the two-tier framework into randomly distributed network topology. In this chapter we again start off from a chained-form network topology, but instead of having the sensors line up in a straight line, we vary the sensor location slightly to make a more irregularly distributed network, and then performed the two-tier framework on it. The results show that the MMSE in this case is on par with the regularly distributed network. The second part of the Chapter 4 results demonstrate how to identify a chained-form network topology over a randomly distributed network topology, and the tracking error was computed as MMSE over time, and with noise level of  $\sigma = 0.1$ , the error was within 0.1 meter, and with noise level of  $\sigma = 0.5$ , the error was around 0.3m with a couple spikes of 0.5m.

The focus of Chapter 5 is the grid-like network topology, in addition to typical tracking result; we analyzed and compared the communication cost (in terms of messages transmitted) against the classic flooding algorithm. Comparing to the flooding algorithm, we have a relatively high overhead at start up, in order to construct the grid-like structure. After the grid-like structure was constructed, searching through the grid-like structure would have a flat  $O(n)$  complexity. The flooding algorithm would have a  $O(n^2)$  complexity throughout all time. In our 100-node network, factoring in the initial high cost we need to setup the grid-like structure, with in 3 time sequences, the message sent by flooding will exceeds the grid-like structure.

## Future Work

As the example provided in Chapter 7, the two-tier framework, and the grid-like network framework can be used to provide location service in various place settings. However the grid-like network framework has much more potential than just for tracking and localization application. Especially for the grid-like network framework introduced in Chapter 5, there are unlimited potential in dynamic topology manipulation.

There are two areas I see as potential follow-up for this thesis. The first area is network topology expansion, since each clique are conditionally independent to each other, the network should be easily split into multiple sub-networks, or multiple small networks can be combined into a single joint network. This area should have a lot of real world applications such as real time traffic control, where vehicles will join or leave the network at random at all time (potential customers can be fleet management companies, group riding events for bike riders etc). These applications will greatly benefit from the dynamic topology management and the pure peer to peer communication.

The second area for future development is self adaptive network. Since in the partitioned network, the cliques are conditionally independent, they are only linked by their communication link instead of correlated by the physical location of the clique. Hence it is possible to let the cliques to make minor adjustments to their physical locations to get a better line of sight, as long as it is still maintained inside the communication range, the topology of the network should not need to change, thus the self-adapted network.

Overall, the two-tier framework and the grid-like network framework introduced in this thesis have great potentials for countless applications, and future researches.

# References

- [1] L. Shi, Z. Zhao, and J. Tan, “Near optimal two-tier target tracking in sensor networks,” in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 1993–1996, Oct 2007.
- [2] L. Shi, J. Tan, and Z. Zhao, “Target tracking in sensor networks using statistical graphical models,” in *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pp. 2050–2055, Feb 2009.
- [3] L. Shi and J. Tan, “Distributive target tracking in sensor networks with a markov random field model,” in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 854–859, Oct 2009.
- [4] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, “Design of a wireless sensor network platform for detecting rare, random, and ephemeral events,” *Proceedings of the 4th international symposium on Information processing in sensor networks*, pp. 497–502, 2005.
- [5] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, and L. Gu, “Energy-efficient surveillance system using wireless sensor networks,” *In Mobisys*, pp. 270–283, 2004.
- [6] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh, “Vigilnet: An integrated sensor network system for energy-efficient surveillance,” *ACM Transaction on Sensor Networks*, vol. 2, pp. 1–38, 2006.
- [7] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. shiuan Peh, and D. Rubenstein, “Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet,” *Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.
- [8] N. Xu, S. Rangwala, and et al, “A wireless sensor network for structural monitoring,” *IN SENSYS*, pp. 13–24, 2004.

- [9] B. Krishnamachari and S. Iyengar, "Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 53, pp. 241–250, 2004.
- [10] T. Clouqueur, K. K. Saluja, and P. Ramanathan, "Fault tolerance in collaborative sensor networks for target detection," *IEEE Transactions on Computers*, vol. 53, pp. 320–333, 2003.
- [11] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy aware wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 19, pp. 40–50, 2002.
- [12] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh, "Optimal energy aware clustering in sensor networks," *special issue of Sensors*, 2002.
- [13] Y. Boykov and D. Huttenlocher, *A Bayesian Framework for Model Based Tracking*. Ithaca, NY, USA: Cornell University, 1999.
- [14] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.
- [15] J. Vermaak, S. Godsill, and P. Perez, "Monte carlo filtering for multi target tracking and data association," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 1, pp. 309–332, 2005.
- [16] S. Oh and S. Sastry, "Tracking on a graph," *Fourth International Symposium on Information Processing in Sensor Networks*, pp. 195 – 202, Mar 2005.
- [17] J. Singh, U. Madhow, R. Kumar, S. Suri, and R. Cagley, "Tracking multiple targets using binary proximity sensors," *Proceedings of the 6th international conference on Information processing in sensor networks*, pp. 529–538, 2007.
- [18] L. Hong, J. Rushing, S. J. Graves, S. Tanner, and E. Criswell, "Real time target tracking with binary sensor networks and parallel computing," *IEEE International Conference on Granular Computing*, pp. 112–117, 2006.
- [19] A. T. Ihler, J. W. F. III, and A. S. Willsky, "Particle filtering under communications constraints," *13th Workshop on Statistical Signal Processing*, 2005.
- [20] X. Sheng and Y.-H. Hu, "Distributed particle filters for wireless sensor network target tracking," *International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 845–848, 2005.
- [21] X. Sheng, Y. H. Hu, and R. Parameswaran, "Distributed particle filter with gmm approximation for multiple targets localization and tracking in wireless sensor network," *Proceedings of the 4th international symposium on Information processing in sensor networks*, pp. 181–188, 2005.

- [22] R. Karlsson and F. Gustafsson, "Monte carlo data association for multiple target tracking," *IEEE Target Tracking: Algorithms and Applications*, vol. 1, pp. 13/1–13/5 vol.1, 2001.
- [23] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for general multiple-target tracking problems," *Decision and Control*, Jan 2004.
- [24] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for multiple-target tracking," *Proceeding of the IEEE International Conference on Decision and Control*, Jan 2005.
- [25] C. S. Jensen, H. Lu, and B. Yang, "Graph model based indoor tracking," *Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pp. 122–131, May 2009.
- [26] X. Liu, G. Zhao, and X. Ma, "Target localization and tracking in noisy binary sensor networks with known spatial topology," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. II–1029–II–1032, 2007.
- [27] K. Ren, K. Zeng, and W. Lou, "Fault-tolerant event boundary detection in wireless sensor networks," *Global Telecommunications Conference*, pp. 1 – 5, Jan 2006.
- [28] K. Ren, K. Zeng, and W. Lou, "Secure and fault-tolerant event boundary detection in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, pp. 354 – 363, Jan 2008.
- [29] B. Krishnamachari and S. Iyengar, "Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE TRANSACTIONS ON COMPUTERS*, vol. 53, pp. 241–250, Jan 2004.
- [30] R. Nowak and U. Mitra, "Boundary estimation in sensor networks: Theory and methods," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 2634, pp. 80–95, Jan 2003.
- [31] B. P. Alex, A. Pothen, and X. Yuan, "A clique tree algorithm for partitioning a chordal graph into transitive subgraphs," *Linear Algebra and its Applications*, 1992.
- [32] N. L. Zhang and L. Yan, "Independence of causal influence and clique tree propagation," *International Journal of Approximate Reasoning*, vol. 19, pp. 335–349, 1997.
- [33] F. R. Bach and M. I. Jordan, "Thin junction trees," *Advances in Neural Information Processing Systems 14*, pp. 569–576, 2001.
- [34] F. V. Jensen and F. Jensen, "Optimal junction trees," *Uncertainty in Artificial Intelligence*, pp. 360–366, 1994.

- [35] R. Kindermann and J. L. Snell, "Markov random fields and their applications," *American Mathematical Society*, 1980.
- [36] H. Kuensch, S. Geman, A. Kehagias, H. Kunsch, S. F. Statistik, S. Geman, and A. Kehagias, "Hidden markov random fields," *Annals of Applied Probability*, 1995.
- [37] Y. Zhang, M. Brady, and S. Smith, "Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm," *IEEE Transactions on Medical Imaging*, vol. 20, pp. 45–57, Jan 2001.
- [38] S. Barker and P. Rayner, "Unsupervised image segmentation using markov random field models," *Pattern Recognition*, Jan 2000.
- [39] S. Li, "Modeling image analysis problems using markov random fields," *Handbook of Statistics*, Jan 2000.
- [40] T. Szirányi, J. Zerubia, L. Czúni, and D. Geldreich, "Image segmentation using markov random field model in fully parallel cellular network architectures," *Real-Time Imaging*, Jan 2000.
- [41] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *Fifth Symposium on Operating Systems Design and Implementation*, pp. 147–163, 2002.
- [42] K. Römer, "Time synchronization in ad hoc networks," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '01, (New York, NY, USA), pp. 173–182, ACM, 2001.
- [43] W.-P. Chen, J. C. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, pp. 258–271, 2004.
- [44] G. yao Jin, X.-Y. Lu, and M.-S. Park, "Dynamic clustering for object tracking in wireless sensor networks," *Third International Symposium on Ubiquitous Computing Systems*, pp. 200–209, 2006.
- [45] M. Walchli, P. Skoczylas, M. Meer, and T. Braun, "Distributed event localization and tracking with wireless sensors," *Proceedings of the 5th international conference on Wired/Wireless Internet Communications*, pp. 247–258, 2007.
- [46] Y. Zou and K. Chakrabarty, "Target localization based on energy considerations in distributed sensor networks," *IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 51 – 58, may 2003.
- [47] E. Olule, G. Wang, M. Guo, and M. Dong, "Rare: An energy-efficient target tracking protocol for wireless sensor networks," *International Conference on Parallel Processing Workshops*, pp. 76 –76, sept. 2007.

- [48] R. Tharmarasa, T. Kirubarajan, A. Sinha, and T. Lang, “Decentralized Sensor Selection for Large-Scale Multisensor-Multitarget Tracking,” 2011.
- [49] Y. E. M. Hamouda and C. Phillips, “Metadata-Based Adaptive Sampling for Energy-Efficient Collaborative Target Tracking in Wireless Sensor Networks,” 2010.
- [50] C. J. C. Jiang, D. Y. D. Yuan, and Y. Z. Y. Zhao, “Towards Clustering Algorithms in Wireless Sensor Networks-A Survey,” 2009.
- [51] H. Kung and D. Vlah, “Efficient location tracking using sensor networks,” *IEEE Wireless Communications and Networking*, vol. 3, pp. 1954 –1961 vol.3, march 2003.
- [52] H.-W. Tsai, C.-P. Chu, and T.-S. Chen, “Mobile object tracking in wireless sensor networks,” *Computer Communication*, vol. 30, no. 8, pp. 1811–1825, 2007.
- [53] S. Tran and T. Yang, “Oco: Optimized communication organization for target tracking in wireless sensor networks,” *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 1, pp. 428 –435, june 2006.
- [54] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky, “Nonparametric belief propagation for self-calibration in sensor networks,” *IEEE Journal of Selected Areas in Communication*, vol. 23, no. 4, pp. 809–819, 2005.
- [55] V. Savic and S. Zazo, “Nonparametric boxed belief propagation for localization in wireless sensor networks,” *Third International Conference on Sensor Technologies and Applications*, pp. 520 –525, june 2009.
- [56] C. Picus, L. Cambrini, and W. Herzner, “Boltzmann machine topology learning for distributed sensor networks using loopy belief propagation inference,” *Seventh International Conference on Machine Learning and Applications*, pp. 344 –349, dec. 2008.
- [57] V. Savic and S. Zazo, “Sensor localization using nonparametric generalized belief propagation in network with loops,” *12th International Conference on Information Fusion*, pp. 1966 –1973, july 2009.
- [58] X. X. Cui, Z. Fang, P. C. Zhou, and K. Liu, “Compact distributed target-tracking algorithm with binary sensors,” *IET Wireless Sensor Systems*, vol. 1, no. 4, p. 218, 2011.
- [59] S. S. Ahmeda, M. Keche, I. Harrison, and M. S. Woolfson, “Adaptive joint probabilistic data association algorithm for tracking multiple targets in cluttered environment,” *IEEE Proceedings of Radar, Sonar and Navigation*, vol. 144, no. 6, pp. 309–314, 1997.

- [60] J. Vermaak, N. Ikoma, and S. Godsill, "Extended object tracking using particle techniques," *IEEE Aerospace Conference*, vol. 3, 2004.
- [61] W. Chen, J. C. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, pp. 258–271, 2004.
- [62] S. Oh and S. Sastry, "Tracking on a graph," in *Fourth International Symposium on Information Processing in Sensor Networks*, pp. 195–202, 2005.
- [63] S. Oh, S. Sastry, and L. Schenato, "A hierarchical multiple-target tracking algorithm for sensor networks," *In Proceeding of the International Conference on Robotics and Automation*, pp. 2197–2202, 2005.
- [64] S. Oh, P. Chen, M. Manzo, and S. Sastry, "Instrumenting wireless sensor networks for real-time surveillance," *Proceeding of the International Conference on Robotics and Automation*, pp. 3128–3133, 2006.
- [65] P. Arambel, J. Silver, J. Krant, M. Antone, and T. Strat, "Multiple-hypothesis tracking of multiple ground targets from aerial video with dynamic sensor control," *In proceeding of Signal Processing, Sensor Fusion, and Target Recognition*, vol. 5429, pp. 23–32, 2004.
- [66] W. Ng, J. Li, S. Godsill, and J. Vermaak, "A hybrid approach for online joint detection and tracking for multiple targets," *IEEE Aerospace Conference*, pp. 2126–2141, 2005.
- [67] M. Vemula, M. F. Bugallo, and P. M. Djuric, "Target tracking in a two-tiered hierarchical sensor network," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp. 969–972, 2006.
- [68] Q. Diao, J. Lu, W. Hu, Y. Zhang, and G. R. Bradski, "Dbn models and a prediction method for visual tracking," *Signal and Image Processing*, pp. 189–194, 2003.
- [69] P. Nillius, J. Sullivan, and S. Carlsson, "Multi-target tracking - linking identities using bayesian network inference," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2187–2194, 2006.
- [70] B. J. Frey and R. Koetter, "Exact inference using the attenuated max-product algorithm," in *Advanced Mean Field Methods: Theory and Practice*, MIT press, 2000.
- [71] M. Ding, D. Chen, K. Xing, and X. Chen, "Localized fault-tolerant event boundary detection in sensor networks," *24th Annual Joint Conference of the IEEE INFOCOM*, Jan 2005.



- [72] A. Shahrokni, T. Drummond, and P. Fua, "Texture boundary detection for real-time tracking," *LECTURE NOTES IN COMPUTER SCIENCE*, Jan 2004.
- [73] B. Manjunath and R. Chellappa, "Unsupervised texture segmentation using markov random field models," *IEEE Transactions on Pattern Analysis and Machine*, Jan 1991.
- [74] A. Savvides, H. Park, and M. Srivastava, "The n-hop multilateration primitive for node localization problems," *Mobile Networks and Applications*, Jan 2003.
- [75] Y. Ghiassi-Farrokhfal and M.-R. Pakravan, "Cross-layer flooding for sensor networks without location information," in *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pp. 5 pp.–114, 2005.
- [76] M. Karzand, M. Rezaie, and V. Mansouri, "Adaptive fault-tolerant data flooding for energy-aware sensor networks," in *Intelligent Sensing and Information Processing, 2005. Proceedings of 2005 International Conference on*, pp. 83–87, 2005.
- [77] R. Farivar, M. Fazeli, and S.-G. Miremadi, "Directed flooding: a fault-tolerant routing protocol for wireless sensor networks," in *Systems Communications, 2005. Proceedings*, pp. 395–399, 2005.
- [78] S. Aly, M. Youssef, H. Darwish, and M. Zidan, "Distributed flooding-based storage algorithms for large-scale wireless sensor networks," in *Communications, 2009. ICC '09. IEEE International Conference on*, pp. 1–5, 2009.
- [79] Karyono, I. Martoyo, H. Uranus, Junita, and B. Kim, "Simulation of gravity and flooding algorithm for wireless sensor network," in *Communications (MICC), 2009 IEEE 9th Malaysia International Conference on*, pp. 927–931, 2009.
- [80] L. Zhao, G. Liu, J. Chen, and Z. Zhang, "Flooding and directed diffusion routing algorithm in wireless sensor networks," in *Hybrid Intelligent Systems, 2009. HIS '09. Ninth International Conference on*, vol. 2, pp. 235–239, 2009.
- [81] H. Jeong, H. Jeong, and Y. Yoo, "Dynamic probabilistic flooding algorithm based-on neighbor information in wireless sensor networks," in *Information Networking (ICOIN), 2012 International Conference on*, pp. 340–345, 2012.
- [82] S.-H. Park, Y.-H. Choi, S.-H. Baek, T.-K. Lee, and S.-Y. Oh, "Feature localization using neural networks for cleaning robots with ultrasonic sensors," in *International Conference on Control, Automation and Systems*, pp. 449–453, oct. 2007.
- [83] S. Knauth, C. Jost, M. Fercu, and A. Klapproth, "Design of an ultrasonic localisation system with fall detection for use in assisted living environments," *IET Seminar Digests*, vol. 2009, no. 12725, pp. 4–4, 2009.

- [84] W. Xiao, J. K. Wu, L. Shue, Y. Li, and L. Xie, "A prototype ultrasonic sensor network for tracking of moving targets," in *IST IEEE Conference on Industrial Electronics and Applications*, pp. 1 –6, may 2006.
- [85] T. Srinath, "Localization in resource constrained sensor networks using a mobile beacon with in-ranging," in *International Conference on Wireless and Optical Communications Networks*, pp. 5 pp. –5, 0-0 2006.
- [86] X. Li, H. Shi, and Y. Shang, "A sorted rssi quantization based algorithm for sensor network localization," in *International Conference on Parallel and Distributed Systems*, vol. 1, pp. 557 – 563 Vol. 1, july 2005.
- [87] T. Hori, Y. Nishida, H. Aizawa, S. Murakami, and H. Mizoguchi, "Sensor network for supporting elderly care home," in *Proceedings of IEEE Sensors*, pp. 575 – 578 vol.2, oct. 2004.
- [88] I.-J. Wang, J. H. Lim, and A. Terzis, "Energy-efficient sensor management in multi-static active sonar networks," in *42nd Asilomar Conference on Signals, Systems and Computers*, pp. 1611 –1616, oct. 2008.
- [89] Y.-H. Cho, S.-P. Choi, W.-Y. Kim, and E.-C. Choi, "Development of sensor network nodes for ultrasonic sensor-driven position system inside buildings," in *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 2, pp. 232 –236, june 2006.
- [90] H. Li, D. Miao, J. Chen, Y. Sun, and X. Shen, "Networked ultrasonic sensors for target tracking: An experimental study," in *IEEE Global Telecommunications Conference*, pp. 1 –6, 30 2009-dec. 4 2009.
- [91] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Operation System Rev.*, vol. 36, no. SI, pp. 147–163, 2002.
- [92] P. Challenges, J. A. Stankovic, Q. Cao, T. Doan, L. Fang, Z. He, R. Kiran, S. Lin, S. Son, R. Stoleru, and A. Wood, "Wireless Sensor Networks for In-Home Healthcare:," *High Confidence Medical Devices, Software, and Systems*, pp. 2–3, 2005.
- [93] A. Fleury, M. Vacher, H. Glasson, J. f. Serignat, and N. Noury, "Data Fusion in Health Smart Home: Preliminary Individual Evaluation of Two Families of Sensors," in *Conference of the International Society for Gerontechnology*, 2008.
- [94] S. Srinivasan, J. Han, D. Lal, and A. Gacic, "Towards automatic detection of falls using wireless sensors.," *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 2007, pp. 1379–82, Jan. 2007.

- [95] G. Brown, “An Accelerometer Based Fall Detector: Development, Experimentation, and Analysis,” *Technical report*, 2005.
- [96] M. Kangas, A. Konttila, I. Winblad, and T. Jämsä, “Determination of simple thresholds for accelerometry-based parameters for fall detection.,” *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 2007, pp. 1367–70, Jan. 2007.
- [97] J. Y. Hwang, J. M. Kang, Y. W. Jang, and H. Kim, “Development of novel algorithm and real-time monitoring ambulatory system using Bluetooth module for fall detection in the elderly.,” *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 3, pp. 2204–7, Jan. 2004.
- [98] T. Zhang, J. Wang, L. Xu, and P. Liu, “Fall Detection by Wearable Sensor and One-Class SVM Algorithm,” *Intelligent Computing in Signal Processing and Pattern Recognition*, vol. 345, pp. 858–863, 2006.
- [99] Y. Lee, J. Kim, M. Son, and M. Lee, “Implementation of accelerometer sensor module and fall detection monitoring system based on wireless sensor network.,” *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 2007, pp. 2315–8, Jan. 2007.
- [100] D. Jiangpeng, B. Xiaole, Y. Zhimin, S. Zhaohui, and X. Dong, “PerFallD: A pervasive fall detection system using mobile phones,” pp. 292–297, Mar. 2010.
- [101] T. R. Hansen, J. M. Eklund, J. Sprinkle, R. Bajcsy, and S. Sastry, “Using smart sensors and a camera phone to detect and verify the fall of elderly persons,” in *European Medicine, Biology and Engineering Conference*, In European Medicine, Biology and Engineering Conference, 2005.
- [102] R. Cucchiara, A. Prati, and R. Vezzani, “A multi-camera vision system for fall detection and alarm generation,” *Expert Systems*, vol. 24, no. 5, pp. 334–345, 2007.
- [103] A. Williams, D. Ganesan, and A. Hanson, “Aging in place: fall detection and localization in a distributed smart camera network,” in *15th international conference on Multimedia*, Proceedings of the 15th International Conference on Multimedia, 2007.
- [104] G. Tian and X. Li, “A method for fast fall detection,” pp. 3619–3623, June 2008.
- [105] Y. Xinguo, “Approaches and principles of fall detection for elderly and patient,” pp. 42–47, July 2008.

- [106] H. Foroughi, A. Rezvanian, and A. Pazirae, “Robust Fall Detection Using Human Shape and Multi-class Support Vector Machine,” pp. 413–420, Dec. 2008.
- [107] G. Diraco, A. Leone, and P. Siciliano, “An active vision system for fall detection and posture recognition in elderly healthcare,” 2010.
- [108] M. Grassi, A. Lombardi, G. Rescio, P. Malcovati, M. Malfatti, L. Gonzo, A. Leone, G. Diraco, C. Distanto, P. Siciliano, V. Libal, J. Huang, and G. Potamianos, “A hardware-software framework for high-reliability people fall detection,” pp. 1328–1331, Oct. 2008.
- [109] C. Doukas and I. Maglogiannis, “Advanced patient or elder fall detection based on movement and sound data,” pp. 103–107, Jan. 2008.

# Appendix

## Appendix A Copyright Permission for Chapter Chapter 3, 4 and 5

The following letter from IEEE gives permission to reprint Chapter 3, Chapter 4 and Chapter 5.

### Thesis / Dissertation Reuse

**The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:**

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.