THE HONG KONG POLYTECHNIC UNIVERSITY
DEPARTMENT OF COMPUTING

RANGE-FREE AND RANGE-BASED LOCALIZATION OF WIRELESS SENSOR NETWORKS

By
Qingjun XIAO

A thesis submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
May 2011

CERTIFICATE OF ORIGINALITY

Date: **May 2011**

Author: **Qingjun XIAO**

Title: **Range-Free and Range-Based Localization of Wireless Sensor Networks**

Department: **Department of Computing**

Degree: **Ph.D.** Convocation: **July 16** Year: **2011**

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

Signature of Author Qingjun XIAO

Abstract

Wireless sensor networks consist of many wireless sensor nodes that enable the collection of sensor data from the physical world. A key requirement to interpreting the data is to determine the locations of the sensor nodes. The localization techniques developed can be divided into two categories: *range-free* and *range-based*.

Range-free localization usually assumes *isotropic networks* where the hop count between two nodes is proportional to their distance. However, *anisotropic networks* are more realistic due to the presence of various anisotropic factors in practice, e.g. irregular radio propagation, low sensor density, anisotropic terrain condition, and obstacles which can detour the shortest path between two nodes. The previous anisotropy-tolerating solutions focused on only one anisotropic factor - the obstacles. We will propose a *pattern-driven localization scheme* to tolerate multiple anisotropic factors.

Range-based localization assumes that the inter-node distances can be accurately measured by special ranging hardware. There are two important issues: (1) the ranging noise which affects the *localization accuracy*; (2) the collinearity of critical node sets which may produce unanticipated flip ambiguities and harm the *localization robustness*. However, the previous research does not fully address these two issues,

especially for patch merging, a powerful tool to localize sparse sensor networks. We will present our *inflexible body merging* algorithm to address these two issues for both patch merging and multilateration. Our algorithm can also improve the percentage of localizable nodes by nearly two times in sparse networks as compared with state-of-the-art work.

Another critical issue for range-based localization is the existence of *outliers* in raw data (i.e. distance measurements and anchor positions) which strongly deviate from their true values. These outliers can severely degrade localization accuracy and need to be rejected. Previous studies have two inadequacies of (1) focusing on adding an outlier rejection ability to multilateration but neglecting patch merging; (2) rejecting only the outlier distances but neglecting outlier anchors which are more difficult to remove, because outlier anchors may *collude* by declaring positions in the same coordinate frame. We will present an algorithm to reject both outlier distances and colluding outlier anchors, in both dense networks and sparse networks.

Publications

JOURNAL PAPERS

1. **Qingjun Xiao**, Bin Xiao, Jiannong Cao, "Iterative Localization of Wireless Sensor Networks: An Accurate and Robust Approach", *submitted to IEEE/ACM Transactions on Networking (TON).*

2. **Qingjun Xiao**, Bin Xiao, Kai Bu, "Robust Localization against Outliers in Wireless Sensor Networks", *submitted to ACM Transactions on Sensor Networks (TOSN).*

3. **Qingjun Xiao**, Bin Xiao, Jiannong Cao, Jianping Wang, "Multihop Range-Free Localization in Anisotropic Wireless Sensor Networks: A Pattern Driven Scheme", *IEEE Transactions on Mobile Computing (TMC)*, vol. 9, no. 11, November 2010, pp. 1592-1607.

4. Bin Xiao, Lin Chen, **Qingjun Xiao**, Minglu Li, "Reliable anchor-based sensor localization in irregular areas", *IEEE Transactions on Mobile Computing (TMC)*, vol. 9, no. 1, January 2010, pp. 60-72.

CONFERENCE PAPERS

1. Kai Bu, Bin Xiao, **Qingjun Xiao** and Shigang Chen, "'Efficient Pinpointing of Misplaced Tags in Large RFID Systems", in *Proceedings of IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, Salt Lake City, Utah USA, June 27-30, 2011.

2. **Qingjun Xiao**, Bin Xiao, Jiaqing Luo, Guobin Liu, "Reliable Navigation of Mobile Sensors in Wireless Sensor Networks without Localization Service", In *Proceedings of IEEE International Workshop on Quality of Service (IWQoS)*, Charleston, SC, USA, 2009.

Acknowledgements

This work would not have been possible without the support of my colleagues, friends, and mentors. Specifically, I would like to thank my supervisor, Bin Xiao, who has given me all the support and guidance I needed as a doctoral student. He is not only a good researcher with broad knowledge but also a nice friend and kind person. I am very grateful to have had his trust in my ability, and I have often benefitted from his insight and advice. I am also thankful to Jiannong Cao, my co-supervisor, for his many suggestions and constant support during this research. He has sharp intuition, and grand insight about research, and taught me a lot about how to be a good researcher.

This thesis was also made possible by collaboration and consultation with many colleagues. I would like to thank Jiaqing Luo, Guobin Liu, Kai Bu, Wei Feng, Weiping Zhu, Xiaopeng Fan, and Lin Zhang at Hong Kong Polytechnic University; Lin Chen at IBM China Research Lab; Hu Guan at Shanghai Jiaotong University; Yong Tang at National University of Defense Technology. Working with them is a valuable experience for me.

Of course, I am grateful to my parents for their patience and love. They have brought me up with their love. They have also taught me a lot about how to be a good person. Without them, this work would never have come into existence. Also,

I wish to offer my thanks to my girlfriend Chen Sujie, for her patience, waiting and love. We have been separated for nearly a year until now. However, she has never complained to me about this separation. Instead, she always encourages me and takes care of my parents.

I should also mention that my graduate studies in Hong Kong were supported in part by the fund from Hong Kong government, including HK PolyU G-U810 and CityU project no. 9681001.

Hong Kong S.A.R., China Qingjun XIAO

August 2, 2011

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background of Wireless Sensor Networks

In the late 1990s, with the declining of manufacturing cost of radios and microprocessors, there emerge *wireless sensor networks* (WSNs) which consist of a large quantity of inexpensive sensor nodes [Culler et al., 2004]. These spatially distributed nodes can monitor environmental conditions (e.g. temperature, sound, vibration,

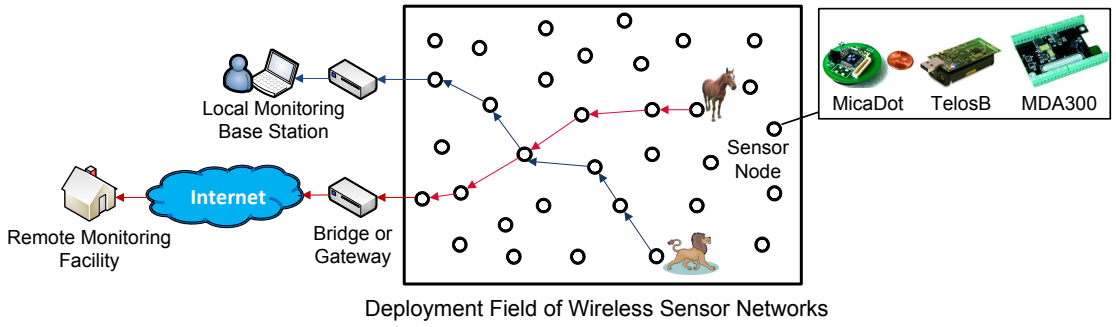
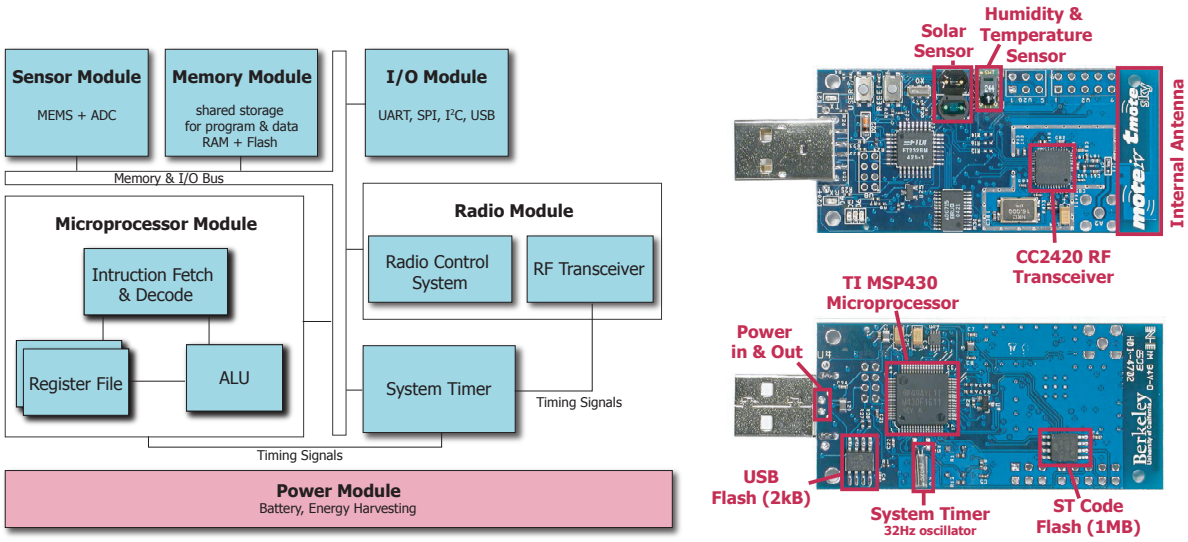


Fig. 1.1: Architecture of Wireless Sensor Networks.

speed or pollutants), collaborate to preprocess the sensed data and relay the data of interest to a base station [Estrin et al., 1999]. Nowadays, WSNs have become

extremely popular due to the wide applications in smart power grid, intelligent transport system, healthcare, home automation, tracking, structural health monitoring, environmental and habitat monitoring. In summary, wireless sensor networks have tremendous potential because they will expand our ability to monitor and interact remotely with the physical world.



(a) Generalized Architecture for Wireless Sensors (b) Front and Back of TMote Sky Mote

Fig. 1.2: Architecture of an Individual Sensor Node

An individual sensor node is essentially an embedded wireless device, which comprised of four basic modules as illustrated in Fig. 1.2(a): the sensor module which can collect sensing readings; the microprocessor module and the memory module can process and store the collected data; the radio module enables the collaboration between sensor nodes via wireless communications. An example is the TMote Sky mote whose components are listed in the following table [Moteiv, 2006] and illustrated in Fig. 1.2(b). Multiple such TMotes can form a mesh network to forward collected data to the base station, utilizing the mesh support from TinyOS [Levis et al., 2008, 2004],

Sensor Module:	Integrated Humidity, Temperature, and Light sensors
Microprocessor Module:	8MHz TI MSP430 microprocessor (10k RAM, 48k Flash)
Radio Module:	250kbps 2.4GHz IEEE 802.15.4 Chipcon RF Transceiver; integrated antenna with range 50m indoors / 125m outdoors

the most popular embedded operating system for wireless sensor nodes.

Wireless sensor networks have attracted extensive research efforts. We briefly list some of the hot research problems as follows, to give readers an overall impression of wireless sensor network research.

- Low Power MAC (Medium Access Control) Protocols: Wireless sensor nodes are energy-constrained embedded devices, whose energy supplies mainly rely on their battery packs. For a sensor node, each component needs to be optimized to save energy cost and prolong the overall operation time. Among the four basic components, the radio module is the most energy-expensive. Pottie and Kaiser [Pottie and Kaiser, 2000] have shown that the energy required to transmit 1 bit over 100 meters, which is 3 joules, can be used to execute 3 million instructions. Therefore, it is not surprising that extensive research efforts have been devoted to designing low power MAC protocols, which can reduce the energy wasted on idle listening, overhearing, resend due to collision, and unnecessary control overhead [Bachir et al., 2010, Buettner et al., 2006, Dutta et al., 2010, Polastre et al., 2004, Rhee et al., 2005, Sun et al., 2008, Ye et al., 2002, 2006].

- Reliable Data Collection: The wireless channel used by wireless motes is lossy by nature, because (1) to save manufacturing cost, the motes are usually equipped with narrow-band and low-power transceivers; (2) the motes are frequently placed directly on ground or in complex indoor environments where multipath

fading becomes a serious problem; (3) the presence of sporadic natural interferences also becomes a serious issue. However, many applications (e.g. industrial process control) require that the collected data be delivered reliably to the base station. Thus researchers have investigated various methods to improve the reliability of data collection, e.g. by exploiting antenna diversity, temporal diversity, spectrum diversity, or network-wide scheduling of transmissions [Lennvall et al., 2008, Pister and Doherty, 2008, Song et al., 2008].

- Sensor Coverage: While the previous two research problems are on the radio component of sensor nodes, the coverage problem is focused on the sensing component of sensor nodes. Its observation is that the major purpose of deploying a sensor network is to monitor a region of interests, e.g. for intrusion detection or intruder tracking. Most of these applications have the same requirement that every point in the service area of the sensor network is covered by at least k sensors, where k is a given parameter [Meguerdichian et al., 2001]. This coverage problem becomes more challenging when it involves with the sleep scheduling of sensor nodes [Kumar et al., 2004], or when the coverage verification algorithm needs to be a distributed one [Huang and Tseng, 2005].

- Clock Synchronization: Time synchronization is a critical piece of infrastructure for any distributed system. For wireless sensor networks, the sensor nodes also need to agree on a common notion of time, as required by data fusion. Traditional synchronization algorithms assume that the entities to synchronize are only one hop away from the time source, e.g. to synchronize a GPRS base station with GPS satellites. However, in wireless sensor networks, the sensor nodes, which need to synchronize against each other, can be multiple hops away.

A research issue is how to achieve a balance between synchronization accuracy and energy consumption in such multihop networks [Sundararaman et al., 2005].

- Data Aggregation and Fusion: The data, which are to be forwarded to the base station, can be preprocessed within the network and thus compressed in volume, which is called data aggregation. Data fusion refers to the concept that sensor nodes cooperate to merge individual sensor readings into a high-level sensing result, such as integrating a time series of position measurements into a velocity estimate.

- Various other problems which are application specific, e.g. smart power grid [Jiang et al., 2009], intelligent transport system, healthcare [Gao et al., 2007, Mandal et al., 2009], home automation, tracking [Smith et al., 2004, Volgyesi et al., 2007], structural health monitoring [Ceriotti et al., 2009], environmental and habitat monitoring [Szewczyk et al., 2004, Werner-Allen et al., 2006].

1.2 Sensor Network Localization Problem

For most WSNs applications and protocols, it is essential to know the locations of the sensor nodes. For environmental monitoring application, the sensed data are meaningless without the location information where the data is obtained. For routing protocol, communication cost can be reduced by geographical routing if node locations are known [Karp and Kung, 2000, Ko and Vaidya, 2000]. Other location-dependent applications are sensing coverage, location directory service [Abraham et al., 2004, Li et al., 2000], tracking (e.g. asset tracking in residential environments or tank tracking in the military background) [Hu and Evans, 2004, Smith et al., 2004].

The system service of WSNs that obtains the critical location knowledge is called *localization service*. It is not an easy task to give location knowledge to sensor nodes in a large-scale network, because the per-node manual configuration is too troublesome and equipping each node with a GPS (Global Positioning System) receiver is over-expensive. Therefore, a plethora of previous studies can be found to automatically derive sensor locations in a network topology, which is known as *network localization* problem [Eren et al., 2004, Goldenberg et al., 2006, Horn et al., 1988, Jin et al., 2011, Li and Liu, 2007, Lim and C., 2005, Moore et al., 2004, Niculescu and Nath, 2003, Priyantha et al., 2003, Savvides et al., 2003b, Shang and Ruml, 2004, Wang et al., 2008, Whitehouse et al., 2005, Zhang et al., 2011].

For the network localization problem, there are two basic assumptions. The first assumption is that the network contains a small number of special nodes with the priori knowledge of their locations, which are called *anchor nodes*. The location knowledge can be acquired from GPS receivers equipped on these anchor nodes, or by manually configuring these nodes with locations to save cost, or by equipping the anchor nodes with laser range finders which can achieve sub-centimeter level accuracy. The purpose of deploying anchor nodes in a network is to guarantee the estimated locations for other unknown nodes are defined in the coordinate frame of anchor nodes, which is well-known to network users. The second assumption is about the topology edges, which can be divided into two categories: *range-based* and *range-free* [He et al., 2003]. The former assumes the length of topology edges can be measured by special ranging hardware which is equipped on each sensor node. In contrast, the latter does not need special ranging hardware and uses the dipole antennas already on sensor nodes to achieve a rough localization.

This dissertation will examine both categories of localization algorithms. More specifically, range-free localization derives locations from the network connectivity (i.e. who is within the radio range of whom) or from the received signal strength (RSS), which can be directly obtained from radio-frequency (RF) transceivers. In contrast, range-based localization assumes the sensor nodes are equipped with extra ranging hardware to measure the geometric relation between neighboring nodes, e.g. their distance or their relative angle. For a WSNs project, the choice of a localization technique (range-free or range-based) depends on the application-specific factors, including the accuracy requirement of intended WSNs application, the needed coverage of sensor fields, and the budget for the whole project.

We briefly list in Table 1.1 the pros and cons of several localization techniques. In range-based localization category, the ultrasound-based technique can provide high precision below 1cm. But the range of its ranging signals is as small as 6m, and these ultrasonic signals can be easily blocked by obstacles like walls [Priyantha et al., 2000, 2001]. The UWB[1]-based technique [Molisch et al., 2004] provides both high sub-meter precision and good coverage of 50m range with the wall-penetration ability which is especially suitable for indoor environments. But UWB is still prohibitively expensive now for low-end users. In contrast, RSS fingerprinting in the range-free category is cost-effective but provides only a coarse accuracy above 3m and needs regular maintenance [Lorincz and Welsh, 2005]. Connectivity-based localization, although providing the lowest accuracy (about one-third of radio range), has the lowest cost by requiring sparser anchor distribution than RSS fingerprinting, and its coarse accuracy can satisfy the requirement of some location-based applications [He et al., 2003].

Table 1.1: Pros and Cons of Range-Free Localization and Range-Based Localization

	Advantages	**Disadvantages**
Range-free	*Cheap Cost* since connectivity or received signal strength (RSS) data can be directly obtained from the RF transceivers already on board	*Low Localization Accuracy* about one third of communication range (≈ 10m); can be improved to around 3m if RSS fingerprinting is used.
Range-based	*High Localization Accuracy* ranging from sub-centimeters to sub-meters depending on the ranging hardware adopted. Generally, sonic methods have higher accuracy than RF methods but have much shorter range.	*High expense* on per-node ranging hardware, e.g. UWB[1] RF TOA [Chung and Ha, 2003], Ultrasonic TOA[2] [Priyantha et al., 2000], RF AOA[3] [Pages-Zamora et al., 2002], Ultrasonic AOA [Priyantha et al., 2001]

1.3 Contributions and Thesis Organization

1.3.1 Range-Free Localization: Robustness against Network Anisotropy

The early range-free localization algorithms function well in isotropic networks assuming the hop count distance between two nodes is proportional to their geometric distance [Doherty et al., 2001, Nagpal et al., 2003, Niculescu and Nath, 2003, Shang et al., 2003]. Their performance deteriorates sharply in concave networks with the presence of obstacles, since the obstacles can detour the shortest path between two nodes and enlarge the anchor-sensor distance estimates. Recently, several methods have been proposed to handle obstacle detours [Li and Liu, 2007, Lim and C., 2005,

[1]Ultra-Wideband (UWB) is a radio technology that can be used at very low energy levels for short-range high-bandwidth communications by using a large portion of the radio spectrum.

[2]Time-of-Arrival (TOA) is a method for determining the distance between two transceivers by measuring the travel time (or round trip travel time) of signals between them.

[3]Angle-of-Arrival (AOA) is a method for determining the direction of propagation of a radio-frequency wave incident on an antenna array.

Shang and Ruml, 2004, Wang and Xiao, 2006].

However, the previous studies focused on only one anisotropic factor, i.e. obstacle detours. Their localization accuracy still degrades in realistic networks with various anisotropic factors, including obstacle detours, irregular radio propagation pattern, non-uniform sensor distribution density, and anisotropic deployment terrain condition with some regions higher than others. The previous methods also have another inadequacy of non-scalability, since their anchor-sensor distance estimates have larger errors with the increase of network scale, which is called error accumulation. Some of the previous methods rely on centralized computation, which consumes the micro sensors' precious energies to collect and disseminate required information. Most of them neglect the impact of last hop distance on the overall distance estimation.

In Chapter 3, we propose a *pattern-driven localization scheme* targeting at large-scale sensor networks with the presence of multiple anisotropic factors [Xiao et al., 2010b]. This scheme can accurately estimate an anchor-sensor distance, even when the shortest path from the anchor to the sensor (1) spans many hops which may incur the accumulation of errors in anchor-sensor distance estimation, (2) has inaccurate per-hop distance due to the interference of various anisotropic factors, and (3) is detoured by obstacles. Our basic idea is that a sensor node can trust the nearby anchors within three hops, and can accurately estimate distances to these *trustworthy anchors* by an isotropic algorithm. For the remote anchors which are suspectable to anisotropic interferences and obstacle detours, the sensor can use the trustworthy nearby anchors as reference stations to revise their distance estimates:

- if there is only one trustworthy nearby anchor, we can revise the remote anchors which are slightly detoured but are interfered by other anisotropic factors;

- if there are two or more trustworthy nearby anchors, we can revise the remote anchors which are both strongly detoured and interfered by other other anisotropic factors.

Our simulation results show that our scheme can improve distance estimation accuracy to be better than $0.4r$, where r is the average radio range. Such a performance is achieved in sparse concave networks with radio anisotropy based on DOI model. In contrast, the state-of-the-art PDM fails in O-shaped networks and cannot handle the last hop distance problem [Lim and C., 2005].

1.3.2 Range-Based Localization: An Accurate and Robust Approach

The previous research on range-based localization advances in two opposite directions: *whole-topology approach* and *iterative approach*. The former directly analyzes the whole network topology [Ji and Zha, 2004, Priyantha et al., 2003]. The latter firstly divides the network topology into small *network elements* (i.e. individual nodes and groups of nodes called patches) and then merges these elements iteratively [Goldenberg et al., 2006, Horn et al., 1988, Meertens and Fitzpatrick, 2004, Moore et al., 2004, Savvides et al., 2003b, Wang et al., 2008]. Recently, the iterative approach becomes more and more popular, because the whole-topology approach is more prone to trap in local minima, especially when applied to concave network topologies with holes inside. In contrast, the iterative approach can avoid the local minima problem because it merges two network elements at one time and handles only a small number of degree-of-freedom during the merging. For example, iterative multilateration [Savvides et al., 2003b] can merge a node into a patch, which has two

degree-of-freedom, i.e. the translations of the node in x and y directions relative the patch. CALL [Wang et al., 2008] proposed to merge two patches, where the right patch has three degree-of-freedom in the coordinate frame of the left patch, i.e. two translations and an additional rotation.

However, the previous iterative localization methods still have three inadequacies. (1) They provided only parts of the sufficient conditions where two network elements can be merged either uniquely or ambiguously. It is desirable to unify these sufficient conditions and give out a more robust condition that also considers previously neglected collinear geometry of sensor nodes. (2) Multilateration can be used to tolerate ranging noise only when merging a node into a patch. The toleration of ranging noise when merging two patches is still not fully addressed. (3) Although RobustQuad can detect the node collinearity for multilateration [Moore et al., 2004], we still need a unified approach to discover node collinearity for network element merging, i.e. for both multilateration and patch merging.

In Chapter 4, we present a body merging (a network element is called a body for short) algorithm to address all the issues above. Our algorithm is both accurate and robust:

- Accuracy and Noise Toleration: align two bodies accurately by finding the best relative position and orientation of the two bodies that can minimize the mean squared error of their constraints. Our basic idea is to model the two bodies to be connected by springs and then relaxes these springs to their minimum energy states. Note that such an optimal alignment can find only one of the possibilities of aligning two bodies.

- Robustness and Collinearity Discovery: enumerate all the possibilities of aligning two bodies. We firstly check various node combinations during body merging and discover the potential collinear geometry in the presence of ranging noise. We then flip one of the body alignment possibilities across the discovered collinear node set and thus obtain other body alignment possibilities.

We also provide a condition for the merging of two bodies ambiguously. The basic idea is that the two bodies should have enough constraints by sharing nodes or being connected by links to confine their continuous relative motions (i.e. Degree-Of-Constraint $\geq$ Degree-Of-Freedom). This condition can unify previous work [Goldenberg et al., 2006, Savvides et al., 2003b, Wang et al., 2008] and help to achieve a higher localization percentage than state-of-the-art CALL [Wang et al., 2008].

1.3.3 Range-Based Localization: Outlier Rejection

Range-based localization needs two types of inputs: the measurements of inter-node distances and the positions of anchor nodes. However, for the measuring of inter-node distances, it is inevitable to have erroneous distance measurements that deviate significantly from true distances (call them *outlier distances* or outlier links). The presence of these outlier distances can be caused by malfunctions of ranging hardware, severe natural interferences to ranging signals, or malicious attacks. For example, ultrasonic TOA may generate outlier distances with enlarged estimates due to non-line-of-sight propagation of sound signals [Whitehouse et al., 2005]. Besides outlier distances, it is inevitable to have *outlier anchors* declaring erroneous locations that significantly deviate from their true locations. The sources of outlier anchors can be misconfigurations or malicious attacks, e.g. Replay attack (i.e. attacker overhears

an anchor location declaration and replays this declaration at other places) and Sybil attack (i.e. attacker compromises an anchor node, exploits its identity and declares erroneous anchor locations at different places [Newsome et al., 2004]).

The outlier distances and outlier anchors can severely degrade the accuracy of localization algorithms. A bunch of algorithms thus are proposed to identify and reject the outliers among normal distance measurements and normal anchors, which is called *robust network localization*. Previous studies in this field are focused on adding outlier rejection ability to multilateration (called *robust multilateration*), because multilateration is a basic operation that can be applied *iteratively* to localize a network [Kiyavash and Koushanfar, 2007, Kung et al., 2009, Li et al., 2005, Liu et al., 2005, Wang et al., 2007]. However, these previous studies have two inadequacies. (1) They are inefficient in sparse networks where the node degree can be six or less, because iterative multilateration becomes weak and even powerless for localization in such networks. We must resort to another operation named patch merging, which however is still vulnerable to outliers. (2) Previous studies neglect the difference between outlier distances and outlier anchors. Outlier anchors are more harmful because multiple outlier anchors may *collude* and declare positions in the same coordinate frame. Their toleration needs global knowledge in a multihop network to exploit the *majority of benign anchors*, i.e. benign anchors outnumber outlier anchors in the whole network.

In Chapter 5, we propose a robust network localization algorithm called *RobustLoc* to reject outlier distances and outlier anchors in networks which can be either sparse or dense. Our RobustLoc algorithm makes the following contributions if compared with previous work. (1) We propose a robust patch merging operation which is

more generalized and powerful than robust multilateration presented in [Kiyavash and Koushanfar, 2007, Liu et al., 2005, Wang et al., 2007]. Our RobustLoc algorithm thus can effectively reject outlier links and meanwhile achieve high localization percentage in sparse networks. In contrast, robust multilateration method can only localize a small proportion of nodes in sparse networks. (2) Our robust patch merging operation handles the non-rejectable outlier distances due to insufficient connectivity in sparse subregions, which is neglected before. (3) Our RobustLoc algorithm can reject both outlier links and outlier anchors. In contrast, a recent paper [Jian et al., 2010] can only reject outlier links, based on the enumeration of realizable generic cycles. (4) Our RobustLoc algorithm can tolerate multiple outlier anchors which may collude under malicious attacks. These declared contributions will be validated by high-fidelity simulations with practical system parameters from [Moore et al., 2004, Savvides et al., 2003b].

Chapter 2

Related Work

In this chapter, we give a brief introduction to related work and further highlight the contributions made in this dissertation.

2.1 Range-Free Localization

Chapter 3 concentrates on the range-free localization, which probably is a much cheaper option than range-based and anchor free localization [Stoleru et al., 2007]. This is because the range-free option leverages the radio transmitter already deployed on each sensor and thus eliminates the need for extra per-node devices or additional infrastructure. However, it is more difficult for the range-free localization to achieve high localization accuracy, which is desirable for location dependent protocols and applications [He et al., 2003]. Therefore, for a range-free solution, accuracy is the most critical factor in deciding its applicability.

A common feature of pioneering range-free solutions (e.g. DV-Hop [Niculescu and Nath, 2003], Amorphous [Nagpal et al., 2003] and MDS-MAP [Shang et al., 2003])

is their assumption about network isotropy. Therefore, their performance degrades severely [Lim and C., 2005] in test beds, where multiple anisotropic factors exist (e.g. concave deployment region, sparse and non-uniform sensor distribution, anisotropic terrain condition and irregular radio pattern). This underperformance has led to extensive research on anisotropy tolerating algorithms. Most of these studies are based on two commonly seen (and also overlapping) assumptions: a fixed number of anchors and the presence of only one anisotropic factor, i.e. obstacle detour.

A fixed number of anchors is assumed by MDS-MAP(P) [Shang and Ruml, 2004] and REP [Li and Liu, 2007]. The algorithms with this assumption share two drawbacks: (1) lack of mechanisms to fully exploit increased anchor density and (2) potential accumulation of error, when the network scale is large. As one example, MDS-MAP(P), following the canonical "divide and conquer" paradigm, splits the network into small overlapping subregions. For each subregion, which is considered to be locally isotropic, MDS-MAP is applied to compute a local map. By merging all these local maps, a global map is formed through a coordinate registration procedure. However, its recursive merge operation is sensitive to the noise in local map construction and suffers from error propagation after several iterations [Meertens and Fitzpatrick, 2004], especially when the network scale is large.

The presence of only one anisotropic factor (i.e., the concave deployment region) is assumed by iMultihop [Wang and Xiao, 2006] and REP [Li and Liu, 2007]. However, the performance of these algorithms may degrade in practice, because the existence of multiple anisotropic factors may be inevitable in test beds and real deployment of WSNs. The iMultihop [Wang and Xiao, 2006] contributes an impressive improvement to the multilateration component of DV-Hop. It is based on the observation that the

shortest path from an anchor to a sensor will deviate far away from straight lines when distorted by obstacles, and the distorted distance estimate is always larger than its real value. Therefore, a set of upper bound quadratic inequality constraints can be added to the MMSE objective function of traditional multilateration. However, this assumption of iMultihop that distance estimates are enlarged due to anisotropy may not hold for networks having multiple anisotropic factors other than obstacle detour. As a summary, the assumption about existence of only one anisotropic factor may weaken the soundness of an algorithm designed to tolerate network anisotropy.

The problem of range-free localization tolerating network anisotropy can be investigated from another perspective by assuming a varied number of anchors proportional to network scale and the presence of multiple anisotropic factors, rather than a fixed number of anchors and the presence of only one anisotropic factor. The PDM [Lim and C., 2005] and our scheme both fall into this category. These two solutions are compared as follows. (1) Our scheme is a distributed solution with less communication overhead. The communication cost of our scheme is $O(MN)$, while that of PDM (a centralized algorithm) is $O(M^2N)$, where M is the number of anchors and N is the number of sensors - Subsection 3.7.5. (2) Our scheme has consistent performance in various shapes (e.g. rectangular, O and U shapes) of sensor deployment fields, but PDM degrades severely in the O-shaped region. (3) Our scheme has higher accuracy than PDM, when sensors are densely distributed, thanks to the ability of CrMcs to handle the last hop distance problem. (4) It is easier to integrate our pattern-driven scheme with the state-of-the-art works in secured localization to additionally tolerate malicious attacks. However, PDM has no such convenience, since (a) if a sensor drops several unreliable anchors individually, then it may become inconvenient for

the sensor to use the global Proximity-Distance-Matrix; (b) if an anchor lies about its own position, the error in this position declaration can pollute all anchor-sensor distance estimates of a sensor, which makes it difficult for GridVoting [Liu et al., 2005] algorithm to detect lying anchors.

2.2 Range-Based Localization: Accuracy and Robustness against Ambiguities

Network localization techniques can be divided into two categories according to *ranging accuracy*.

(1) *Coarse-grained* techniques have low ranging accuracy of meters or tens of meters since they exploit radio attenuation for ranging [He et al., 2003, Li and Liu, 2007, Lim and C., 2005, Wang and Xiao, 2006, Xiao et al., 2010a,b]. They have two advantages of low cost with no requirement for extra ranging hardware and the ability to satisfy accuracy requirements of certain location-dependent protocols and applications [He et al., 2003].

(2) *Fine-grained* techniques can provide high accuracy at centimeter level, which is based on TOA techniques e.g. ultrasonic TOA or round-trip ultra-wideband TOA [Goldenberg et al., 2006, Horn et al., 1988, Moore et al., 2004, Priyantha et al., 2003, Savvides et al., 2003b, Shang and Ruml, 2004, Wang et al., 2008]. The price for this high accuracy is to deploy per node a ranging device.

Chapter 4 focuses on fine-grained localization, in which the inter-node distances can be accurately measured. From the distance measurements, node locations need

to be derived, which can be affected by the following three basic factors:

(1) *Ranging Noise*. Ranging noise if not properly handled can reduce localization accuracy. Multilateration [Foy, 1976] focuses on how ranging noise can be tolerated when aligning a node with a patch. Patch stitching [Horn et al., 1988] solves how ranging noise can be tolerated when merging two patches that share at least three nodes. However, ranging noise toleration is still a problem when merging two patches that share two nodes or less. Our body merging optimization algorithm can tolerate ranging noise no matter how many nodes two bodies share.

(2) *Flip Ambiguities due to Node Collinearity*. Nodes can be roughly collinear when ranging noise exists, and such implicit collinearity can cause unanticipated flip ambiguities which corrupt localization computations. RobustQuad [Moore et al., 2004], which is proposed to avoid unanticipated flip ambiguities, is only applicable to multilateration. But our algorithm can enumerate flip ambiguities for both multilateration and patch merging.

(3) *Network Connectivity*. Connectivity, which means which node pairs have distance measurements, can affect the localizability of a network. As uncovered by global rigidity related researches [Eren et al., 2004, Goldenberg et al., 2006], sensors or patches can be localized if they have sufficient connectivity to already localized nodes. The drawback of the previous studies is that they neglect the case where two patches, if having enough connectivity, can be merged to generate a larger patch. This larger patch has a better chance to find enough connectivity to localized nodes. Although CALL [Wang et al., 2008] can also merge two patches, it neglects the cases where two patches share two nodes (or share one node and are connected by a link). Our inflexible body merging condition can support all the cases and achieve a higher

localization percentage than CALL.

2.3 Range-Based Localization: Outlier Rejection

Chapter 5 focuses on fine-grained localization, in which the inter-node distances can be accurately measured. From the distance measurements, node locations need to be derived. In this field, most of the solutions are based on two basic operations, i.e. multilateration [Goldenberg et al., 2006, Priyantha et al., 2003, Savvides et al., 2003b] and patch merging [Horn et al., 1988, Moore et al., 2004, Shang and Ruml, 2004, Wang et al., 2008]. However, the accuracy of these two basic operations are under the threat of outlier links and outlier anchors. These outliers are inevitable due to hardware malfunctions, natural interferences and malicious attacks.

A school of researchers acknowledge the threat of outliers and try to enhance multilateration with outlier rejection ability, which is called "robust multilateration". Ring-overlapping method [Liu et al., 2005] compares distance measurements to rings, finds the region that is most heavily overlapped by rings, and regards the centroid of this region as a trustworthy location estimate. SISR [Kung et al., 2009] is based on weighted multilateration and assigns smaller weights to outlier links than to normal links, because intuitively outlier links have larger deformation and can be identified. LMS [Li et al., 2005] supports robust multilateration based on least median of squares. $C(n, 3)$ methods [Kiyavash and Koushanfar, 2007, Wang et al., 2007] find an outlier-free link group by checking each group of three links. Different from the previous researches on robust multilateration, we provide a robust patch merging operation that can reject outliers for both patch merging and multilateration. Based on this robust operation, we further solve the problem of robustly localizing a network which

can be either sparse or dense.

A recent work [Jian et al., 2010] proposes an outlier link rejection algorithm which can remove outlier links before the execution of a network localization algorithm. This work identifies outlier links based on the enumeration of realizable generic cycles: a generic cycle is outlier-free if it can be realized, and a link is identified as an outlier if it is not contained in any outlier-free generic cycles. Different from [Jian et al., 2010], we reject outlier links based on robust patch merging operation, and we can additionally reject outlier anchors even when multiple outlier anchors collude due to malicious attacks.

Chapter 3

Range-Free Localization in Anisotropic Wireless Sensor Networks: A Pattern-Driven Scheme

3.1 Overview

In recent years, by the advances in MEMS and communication theory, wireless sensor networks (WSNs) have revealed great potential to provide economical and practical solutions for both civilian and military applications, e.g. tracking, surveillance and environmental monitoring. In many of these applications, knowledge about sensors' geometrical positions is assumed to be an integral part of sensor readings,

and it is also critical for many network protocols, including topology control, clustering and geographical routing. It thus becomes one of the fundamental issues in WSNs to acquire sensor position knowledge, called sensor localization problem.

To address this localization problem, extensive research has been conducted on multihop solutions for the following reasons. It is a naive solution to have all the sensor nodes equipped with GPS receivers to directly contact satellites, because this "one-hop" approach is prohibited by the size, cost and power consumption constraints of sensor nodes. As a compromise, only a small portion of nodes named *anchors* have GPS receivers (or other localization equipments like laser range finder) and can know their positions accordingly, and these anchors can help to locate other "unknown" sensors. The challenging part of this anchor based approach is that the anchors can only be sparsely distributed, in order to reduce WSNs deployment cost. Therefore, the anchors only one hop away from a sensor may not provide enough anchor-sensor distance estimates to localize this sensor. For this reason, researchers actively seek for the multihop localization solutions that can measure anchor-sensor distances spanning multiple hops.

Among various multihop solutions, people pay great attention to the multihop range-free solutions [Doherty et al., 2001, He et al., 2003, Nagpal et al., 2003, Niculescu and Nath, 2003, Shang et al., 2003, Xiao et al., 2008] that utilize only connectivity information, i.e. who is within the radio range of whom. This is because range-free solutions have no requirements for expensive ranging devices and can satisfy the accuracy requirement of many location-based applications [He et al., 2003]. Moreover, current ranging techniques (e.g. TDoA, AoA and RSSI) have their inadequacies [Savvides et al., 2001].

Although the previous multihop range-free solutions [Doherty et al., 2001, He et al., 2003, Nagpal et al., 2003, Niculescu and Nath, 2003, Shang et al., 2003, Xiao et al., 2008] function well in isotropic networks (that assume hop count distance between two nodes is proportional to their geometric distance), their performance deteriorates sharply in anisotropic networks. Network anisotropy stems from various factors, e.g. concave networks, irregular radio propagation pattern, non-uniform sensor distribution density, and anisotropic deployment terrain condition with some regions higher than others. To tolerate these anisotropic factors, several methods [Li and Liu, 2007, Lim and C., 2005, Shang and Ruml, 2004, Wang and Xiao, 2006] have been proposed recently. These methods however have the inadequacy to focus on only one anisotropic factor, like obstacle detour. They may also have the inadequacy of non-scalability due to the error accumulation along with the increase of network scale. Some of these methods rely on centralized computation, which consumes the micro sensors' precious energies to collect and disseminate required information. Most of them neglect the impact of last hop distance on the overall distance estimation.

We focus on multihop range-free localization in anisotropic networks, and propose a distributed pattern-driven scheme to produce accurate estimates of anchor-sensor distances with the presence of multiple anisotropic factors. This accurate distance estimation is the basis of accurate sensor location estimation. The main idea of our pattern-driven scheme is to exploit the observation that the *hop count field* of an anchor (i.e. hop count distribution of sensors with respect to that anchor) can exhibit multiple patterns in an anisotropic network. One example of this coexistence of multiple patterns can be found in Fig. 3.1. As illustrated, region I is within a few hops from the anchor, and the hop count field there approximately exhibits a

Concentric Ring (CR) pattern, in which sensors can approximately treat this anchor as an isotropic anchor. However, region II is far away from the anchor, and the hop count field there exhibits a *Centrifugal Gradient* (CG) pattern, in which sensors can witness the anchor as an anisotropic anchor. CR and CG patterns are different because CG pattern permits the *HopSize* (i.e. average per-hop-distance) to vary in an unpredictable manner due to the disturbance of multiple anisotropic factors, e.g. non-uniform sensor distribution, irregular radio propagation, anisotropic terrain condition. CR and CG patterns however have a shared feature that a rough match is preserved between the *hop count field gradient* (with the greatest rate of increase of hop count) and the *centrifugal direction* (that departs from the anchor). The worst case is that in region III, hop count field exhibits *Distorted Gradient* (DG) pattern, in which the line-of-sight rule is violated by obstacle detour and the field gradient strongly deviates from the centrifugal direction.

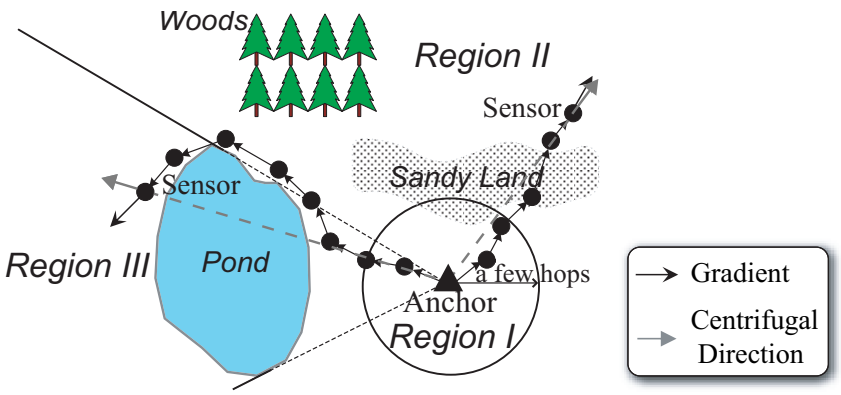


Fig. 3.1: Coexistence of multiple patterns in the hop count field of an anchor.

To achieve accurate distance estimation in this anisotropic sensor network, we assume each sensor has the ability to classify heard anchors into three categories according to the CR, CG, DG patterns. In practice, we put into CR category all the

anchors within three (or four) hops. This threshold is chosen based on our observations of the error accumulation trend (with increased hop count) of the anchor-sensor distance estimation by isotropic algorithms (like Amorphous and DV-Hop) in rectangular anisotropic networks. It is a difficult problem to differentiate between the CG pattern (anisotropic but *slightly* detoured) and the DG pattern (anisotropic and *strongly* detoured). In fact, it may be impossible in practice to accurately recognize the slightly detoured anchors from strongly detoured (or even from moderately detoured anchors), without the global knowledge on network topology, e.g. network boundary and obstacles shapes. A practical and efficient way is to select the eight nearest anchors into the CG category, which thus may contain detoured anchors. We choose this threshold of eight, because from the localization perspective, eight anchors are sufficient to mitigate bad anchor geometry and obtain accurate location estimate. Those anchors that are not eight nearest are put into the DG category.

The three categories of heard anchors corresponding to the three patterns (namely CR, CG, DG) have different dominating error sources in anchor-sensor distance estimation. For the three different categories, we therefore propose different anchor-sensor distance estimation algorithms. (1) For the CR pattern, the last hop distance is an important factor interfering the distance estimation accuracy. To reduce its impact, we propose an algorithm named CrMcs to achieve higher accuracy than DV-Hop [Niculescu and Nath, 2003] and Amorphous [Nagpal et al., 2003]. (2) For the CG pattern, varying *HopSize* becomes the dominating factor and we propose the DiffTriangle to tolerate the inaccurate *HopSize* estimates. The main idea of DiffTriangle is to revise the anchor-sensor distance estimates with the assistance from the nearest anchor to the sensor (namely *Reference Station*), which exhibits the CR

pattern. Because it is inevitable for the CG category to contain detoured anchors, we enhance the DiffTriangle by DiffTriangle* to tolerate obstacle detour additionally. DiffTriangle* however requires two reference stations exhibiting the CR pattern. As a summary, for the CG category, when there are two reference stations in the CR category, DiffTriangle* is adopted; when only one is available, DiffTriangle is used as a backup; when there are none, CrMcs is the only choice left. (3) The surplus anchors in the DG category vulnerable to obstacle detour are dropped. Finally, when sufficient (more than 6) distance estimates are collected from anchors exhibiting the CR or CG pattern, a sensor can deduce an estimate about its own location using weighted MMSE multilateration [Foy, 1976]

Extensive theoretical analysis about estimation accuracy of our pattern-driven scheme can be found in this chapter. We show that, for the CR pattern, CrMcs can effectively suppress distance estimation error below $0.2r$ (r is the average communication range of sensors) when network density is higher than eight. Benefiting from these accurate estimates by CrMcs, distance estimation accuracy of DiffTriangle is improved to be better than $0.4r$, when DiffTriangle is applied to the CG pattern. We demonstrate by simulations the averaged accuracy of DiffTriangle* to be better than $0.5r$ when handling obstacle detour. With these accurate anchor-sensor distance estimates (even when network density is as sparse as eight), the average localization accuracy approaches $0.4r$ according to Cramér-Rao lower bound [Savvides et al., 2003a]. This localization accuracy can satisfy the needs of many location dependent applications, including geographical routing and tracking [He et al., 2003].

Our pattern-driven localization scheme differs from other anisotropy tolerating methods in several fundamental aspects.

- Higher Localization Accuracy:

  1. In the CR pattern, CrMcs minimizes the impact of the last hop distance on distance estimation, which is neglected by other anisotropy tolerating methods.

  2. In the CG pattern, DiffTriangle can effectively tolerate the variation of *HopSize* disturbed by multiple anisotropic factors and DiffTriangle* can additionally tolerate obstacle detour. As a comparison, most of the existing works can tolerate only one anisotropic factor (obstacle detour) and ideally assume circular radio model, dense and uniform sensor distribution and uniform terrain condition.

  3. For the DG pattern, anchors in the DG category are dropped. Therefore, our scheme can easily integrate the state-of-the-art works in secured localization, which can recognize outliers and place them into the DG category to eliminate their impact.

- Less Communication Overhead: The communication overhead of our method is $O(MN)$, while that of PDM is $O(M^2N)$, where M is the number of anchors and N is the number of sensors (including the anchors).

- Reduced Computational Complexity: The arithmetic operations needed by our scheme includes only basic trigonometric functions, bisection root finding and MMSE multilateration [Foy, 1976]. There is no need for inversion of large matrices as in PDM [Lim and C., 2005] and MDS-MAP [Shang and Ruml, 2004].

- Enhanced Algorithm Robustness to Different Network Topologies: Our scheme

is a distributed solution that functions well in all simulated deployment regions, including rectangular, U-shaped, O-shaped regions. Performance of centralized algorithms (like PDM) degrades dramatically in the O-shaped regions (see Subsection 3.7.5).

The rest of this chapter is organized as follows. In Section 3.2, we propose a localization framework to give an overall impression of our pattern-driven localization scheme. Section 3.3 presents CrMcs for distance estimation in the CR pattern to minimize the impact of the last hop distance. Section 3.4 proposes DiffTriangle algorithm to tolerate the varying *HopSize* in the CG pattern. Section 3.5 first quantifies the impact of DG pattern on the accuracy of DiffTriangle and then enhance the DiffTriangle by DiffTriangle* to further tolerate obstacle detour. Section 3.6 provides the algorithm pseudo code to reproduce our simulation results. Section 3.7 presents simulation results, comparing our algorithm with Amorphous and PDM. Finally, we conclude this chapter in Section 3.8.

3.2 Proposed Localization Framework

We present a pattern-driven localization framework, which is deployed at each sensor to estimate its location. The design of this framework is inspired by the fact that from the perspective of an sensor, hop count fields of different anchors may exhibit different patterns, disturbed by different anisotropic factors. To classify heard anchors according to the three patterns (CR - isotropic, CG - anisotropic but slightly detoured, DG - strongly detoured) and to invoke suitable distance estimation algorithms for different patterns, we present the localization framework in Fig. 3.2,

which depicts both the data flow and control flow. We hope the following description can help foster an overall understanding of our pattern-driven localization scheme.

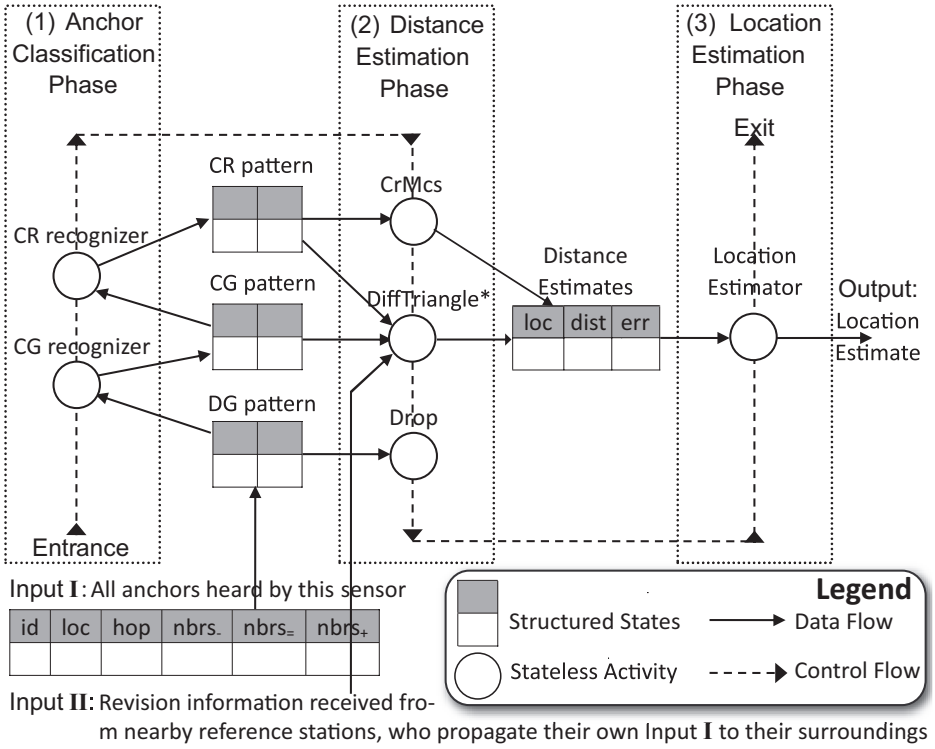


Fig. 3.2: The unified localization framework deployed at each sensor for range-free localization.

From the perspective of control flow, our framework has only one control thread, which consists of three consecutive phases: (1) Anchor Classification Phase; (2) Distance Estimation Phase; (3) Location Estimation Phase. In phase (1), we classify the anchors heard by a sensor into three categories (CR, CG or DG patterns). In phase (2), we schedule a corresponding anchor-sensor distance estimator for each anchor category, which prepares a sufficient number of distance estimates for the next phase. In phase (3), we deduce an estimate about the sensor's location by weighted multilateration.

From the perspective of data flow, the framework has one output (the final location estimate) and two inputs (Input I & II). Firstly, to obtain the Input I (stored temporally in the DG category), each anchor initiates network-wide flooding and each sensor hearing the flooding records the information about the anchor, such as its identifier, location and hop count. Secondly, each sensor locally broadcasts its incomplete Input I (the hop counts to all heard anchors) to its immediate neighborhood, since the Input I additionally requires the knowledge on the number of neighbors having smaller (equal, larger) hop counts than the sensor itself. Finally, each anchor propagates its Input I to three (or four) hops neighborhood by a confined flooding, which is recorded by nearby sensors as their Input II and is needed by our DiffTriangle* algorithm. Therefore, we regard our localization scheme as a distributed solution with only one simple communication protocol - the confined flooding, since the network-wide flooding and the local broadcast are both special cases of the confined flooding.

In phase (1), we classify the set of all heard anchors into three subsets (namely CR, CG, DG) by a sequential execution of two pattern recognizers. Firstly, the CG recognizer relocates undetoured anchors from the DG category into the CG category, since the CG pattern in contrast to the DG pattern has the slight detour assumption. This chapter configures the CG recognizer to trust the nearest 8 anchors, which reduces the chance to contain strongly detoured anchors in the CG category. Secondly, the CR recognizer relocates anchors exhibiting the CR pattern from the CG category into the CR category, since the CR pattern additionally assumes network isotropy (with identical *HopSize*) compared with the CG pattern. For the CR recognizer, we use an empirical rule to trust anchors within a few hops (i.e. three or four hops adjustably).

In phase (2), we adopt different anchor-sensor distance estimators for different

anchor categories (CrMcs for the CR pattern, DiffTriangle* and DiffTriangle for the CG pattern) and drop the remaining anchors in the DG pattern. CrMcs can solve the last hop distance estimation problem for isotropic anchors and it is discussed in Section 3.3. DiffTriangle can tolerate varying *HopSize* and we present DiffTriangle in Section 3.4. DiffTriangle* is an enhancement to DiffTriangle to additionally tolerate obstacle detour and we cover it in Section 3.5. In phase (3), we adopt the weighted multilateration [Foy, 1976] as the location estimator, which is described in Subsection 3.5.2.

An advantage of our framework is its flexibility, which permits the "Stateless Activities" in Fig. 3.2 to be replaced by other algorithms. One example is to replace CrMcs by MDS-MAP [Ji and Zha, 2004], which however incurs higher communication overhead. Another example is to use the GridVoting [Liu et al., 2005] method as the CG recognizor, which can filter both obstacle detour and malicious attacks. However, secured localization is not the focus of this chapter and we find in practice that GridVoting can only detect strongly biased anchors but not moderately detoured anchors. It may be impossible to completely rule out detoured anchors in the CG category, and DiffTriangle* is still necessary even when the GridVoting method is employed. As a summary, our framework has the flexibility to accommodate different algorithm combinations, which can facilitate future studies in sensor network localization.

3.3 CR Pattern and Proposed CrMcs

In this section, we present the CrMcs algorithm to estimate the multihop anchor-sensor distance for isotropic anchors approximately exhibiting the Concentric Ring (CR) pattern. The CR pattern applies in isotropic networks and isotropic regions (like

the region I in Fig. 3.1), in which the field gradient roughly matches centrifugal direction and *HopSize* is identical in all directions as illustrated by Fig. 3.3. We firstly identify the last hop distance problem - an important factor interfering the distance estimation in the CR pattern, and then show the ineffectiveness of DV-Hop and Amorphous to handle this problem. In order to minimize its impact, CrMcs is proposed to reduce distance estimation error of the CR pattern to below $0.2r$ (when network density is above 8). We give out this accuracy bound by theoretical analysis and this accuracy is crucial to guarantee satisfactory performance of DiffTriangle, which will be discussed in the next section. Additionally, the symbols in this section are summarized in TABLE 3.1 and they are also used throughout the chapter.

3.3.1 CR Pattern and The Last Hop Distance

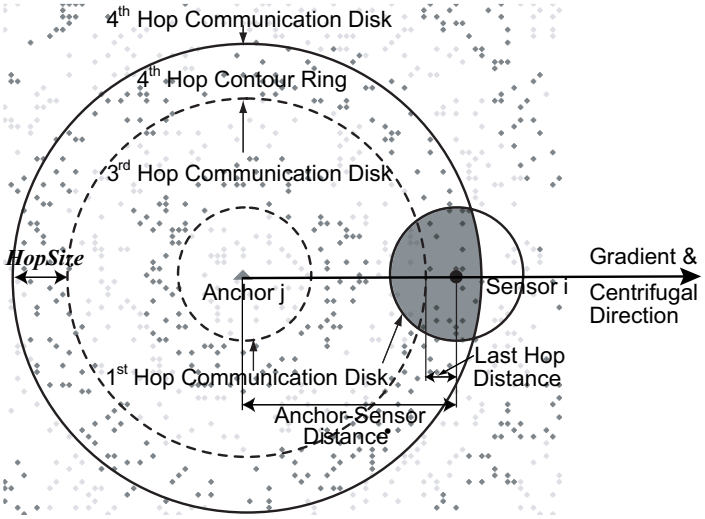


Fig. 3.3: Multihop distance estimation based on the concentric rings pattern.

It is well known that in networks where sensors are uniformly distributed, the hop count field of an anchor approximately demonstrates the Concentric Ring (CR) pattern, under the assumption that the RF transmitters of wireless sensor nodes

have a rotationally symmetric range. We argue that when estimating anchor-sensor distances using the CR pattern, another important factor influencing estimation accuracy (besides the accuracy in *HopSize* estimation) is the last hop distance.

Definition 1 (Definition of Last Hop Distance)**.**
When $h_j(i) = 1$, *the last hop distance of sensor* i *is* $d_j(i)$*;*
When $h_j(i) > 1$, *the last hop distance of sensor* i *is the shortest distance from the contour ring with hop* $h_j(i) - 1$ *to the sensor* i*.*

If there is no methods to effectively estimate the last hop distance, the average distance estimation error will surely exceed one quarter of *HopSize* ($\approx \frac{1}{4} \cdot 0.8r = 0.2r$). This is because the maximum estimation error is half of *HopSize*, since all sensors in a contour ring (like the 4^{th} hop contour ring in Fig. 3.3) have the same hop count and thus share the same anchor-sensor distance estimate that is the arithmetical mean of the inner radius and the outer radius of the ring.

Although this problem of last hop distance is important, it is inappropriately handled by traditional distance estimation algorithms, including DV-Hop [Niculescu and Nath, 2003] and Amorphous [Nagpal et al., 2003]. The DV-Hop, using the following equation, neglects this problem.

$$\hat{d}_{1j}(i) \quad = \quad h_j(i) \cdot d_{hop}$$

The Amorphous, though striving to mitigate the impact of this problem using a method called "smoothing", produces biased distance estimates for the first two hops even when *HopSize* is accurately known, according to the following analysis. Amorphous adopts the smoothed hop count $\overline{h}_j(i)$, rather than the raw hop count $h_j(i)$, to

derive distance estimate $\hat{d}_{2j}(i)$.

$$\hat{d}_{2j}(i) \quad = \quad \overline{h}_j(i) \cdot d_{hop} \tag{3.1}$$

The smoothed hop count $\overline{h}_j(i)$ is calculated by a local averaging around sensor i's immediate neighborhood.

$$\overline{h}_j(i) \quad = \quad \frac{1}{|N(i)|} \sum_{l \in N(i)} h_j(l) - 0.5 \tag{3.2}$$

We analyze the systematic error of Amorphous incurred by the last hop distance in Appendix A.A and plot its result in Fig. 3.4, which indicates Amorphous can

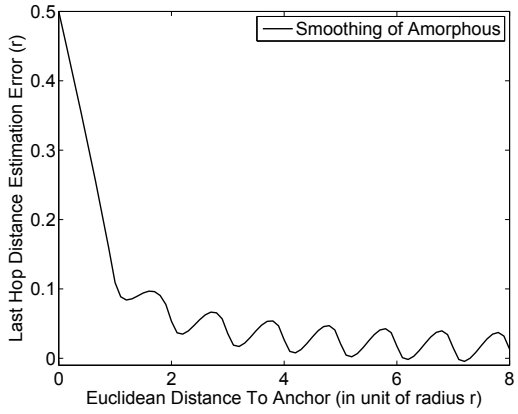


Fig. 3.4: Systematic error of smoothing technique in Amorphous.

be inaccurate in the first two hops. A similar conclusion can be drawn from the simulation results in Section 3.7.

3.3.2 Proposed CrMcs

We propose the CrMcs algorithm to achieve accurate anchor-sensor distance estimation in the CR pattern. The main idea underlying CrMcs is to minimize the

impact of the last hop distance by exploiting the uniform sensor distribution around a sensor's immediate neighborhood (i.e. essentially Monte Carlo sampling). Therefore, from the percentage of neighbors with no larger hop count than the sensor itself, we can estimate the area of the gray region illustrated in Fig. 3.3. With this area estimate, we can derive an anchor-sensor distance estimate that includes the last hop distance.

The first step of CrMcs is to estimate the area of the mentioned gray region, which is the intersected region of anchor j's $h_j(i)$ hop disk - $dsk_j(h_j(i))$ - and sensor i's one hop disk $dsk_i(1)$. We represent its area by $a_j(i)$ and estimate the area by the following equation.

$$\frac{a_j(i)}{\pi r^2} \approx \frac{|N_j(i)|}{|N(i)|} \tag{3.3}$$

Its basic idea is the Monte Carlo sampling, treating each node around sensor i's neighborhood as an independent sampling. In this way, the ratio of $a_j(i)$ to the area πr^2 of sensor i's neighborhood $dsk_i(1)$ can be approximated to the proportion of sensor i's neighbors with hop counts equal to or lower than the sensor i's hop count $h_j(i)$.

The second step of CrMcs is to estimate the radius of the two intersecting disks - $dsk_j(h_j(i))$ and $dsk_i(1)$ by Eq. (3.4), assuming the CR pattern.

$$r(h) \approx (h-1) \cdot d_{hop} + r \tag{3.4}$$

In this equation, the radius of $dsk_j(h_j(i))$ is equal to $r(h_j(i))$ and radius of $dsk_i(1)$ is $r(1)$. In Eq. (3.4), we intentionally assign the one hop disk's radius $r(1)$ as the

sensors' average radio range r, which is determined when the sensors are deployed. This assignment improves estimation accuracy of $r(1)$, while it brings error to estimation of $r(h)$, when $h > 1$. This error is negligible in simulation, since it is quite difficult to estimate *HopSize* (or d_{hop}) precisely in practice due to the existence of radio irregularity (enlarged by the long range link) and it is always underestimated by Kleinrock's equation [Kleinrock and Silvester., 1978].

The final step is to estimate the anchor-sensor distance $d_j(i)$ by Eq (3.5), as the distance between centers of two disks $dsk_j(h_j(i))$ and $dsk_i(1)$.

$$\hat{d}_{3j}(i) \quad = \quad \mathcal{A}^{-1}[r(1),\, r(h_j(i)),\, a_j(i)] \tag{3.5}$$

This equation assumes that the we have the estimates about the radius of the two disks and the area of their intersected region in the previous two steps . The function $\mathcal{A}^{-1}$ is the inverse function of $\mathcal{A}$ in Eq. (3.6), which is established by applying bisection root finding algorithm to function $\mathcal{A}$.

$$\begin{aligned}
a &= \mathcal{A}\,(r_1,\, r_2,\, d) = r_1 \cdot \frac{\theta_1 - sin\theta_1}{2} + r_2 \cdot \frac{\theta_2 - sin\theta_2}{2} \\
\theta_1 &= 2\arccos\frac{x_1}{r_1} \qquad x_1 = \frac{d}{2} + \frac{r_1^2 - r_2^2}{2d} \\
\theta_2 &= 2\arccos\frac{x_2}{r_2} \qquad x_2 = \frac{d}{2} + \frac{r_2^2 - r_1^2}{2d}
\end{aligned} \tag{3.6}$$

The $\mathcal{A}\,(r_1,\, r_2,\, d)$ is a geometric function calculating the area of the intersected region of two disks and taking three parameters - the radius r_1, r_2 of the two intersecting disks and the distance d between their centers. The symbols used in Eq. (3.6) are illustrated by Fig. 3.5.

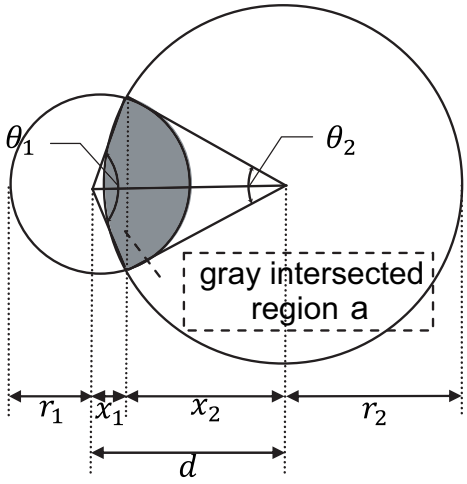


Fig. 3.5: Calculating the area of the intersected region of two disks.

3.3.3 Error Characteristics of CrMcs

Sensor density is an important factor influencing the accuracy of CrMcs, which directly decides the accuracy of the intersected area estimation in Eq (3.3). For this issue, we provide an analysis in Appendix A.B and plot the analysis result in Fig. 3.6, showing that the accuracy of CrMcs is better than $0.2r$ when sensor density > 8. This analysis result is consistent with the simulation result in Section 3.7. The

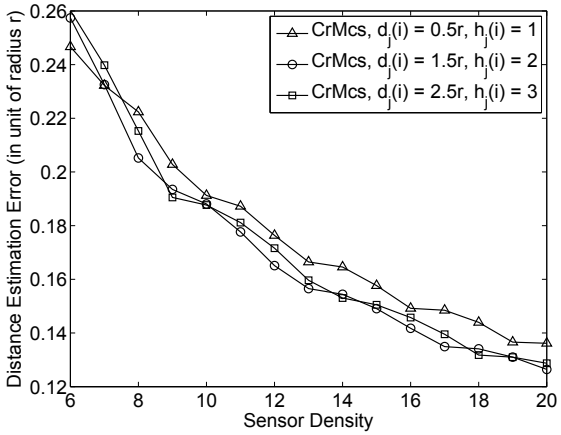


Fig. 3.6: Accuracy of CrMcs in different sensor densities.

limitation of this analysis however is its assumption of accurate $HopSize$ (simplified to r) and it thus is only valuable for low hop count cases, in which the impact of inaccurate $HopSize$ on the accuracy of Eq (3.4) is minimized. For large hop count cases, it is inevitable for the accuracy of CrMcs (and Amorphous) to degrade with the increase of hop count (namely *error accumulation*), since the inaccuracy in *HopSize* get amplified by large hop count in Eq (3.4). It is the topic of the next section on how to tolerate this error accumulation due to varied and inaccurate *HopSize*. Additionally, the accuracy of CrMcs with the presence of radio irregularity (based on DOI model [He et al., 2003]) has been investigated by simulations in Subsection 3.7.3.

3.4 CG Pattern and Proposed DiffTriangle

In this section, we consider the distance estimation problem in anisotropic networks with the presence of various anisotropic factors apart from large obstacles. In this type of networks and regions (like the region II in Fig. 3.1), the hop count fields exhibit the Centrifugal Gradient (CG) pattern. The CG pattern, compared with the CR pattern,

- relaxes the assumption about identical *HopSize* to permit it to vary in all directions - the anisotropic assumption,
- but it preserves the assumption about the rough match between gradient and centrifugal direction - the slight obstacle detour assumption.

Therefore, isotropic algorithms assuming the CR pattern (like DV-Hop and CrMcs) encounter performance degradation in the CG pattern. To tolerate the varying *HopSize*, we propose DiffTriangle algorithm that can reduce the distance estimation error

below $0.4r$ (sensor density ≥ 8). We also provide theoretical analysis for this claimed accuracy in this section.

3.4.1 CG Pattern with Slight Obstacle Detour

Effective localization remains a problem in sparse networks where the sensor density falls in the range of 6 to 15. There are two reasons for using this range. First, Nagpal suggests in [Nagpal et al., 2003] that 15 is a critical minimum sensor density for Amorphous to obtain good accuracy. Second, Kleinrock and Silvester prove in [Kleinrock and Silvester., 1978] that 6 is the optimum sensor density to maintain the network connectivity. The localization problem in sparse networks deserves investigation, because lower sensor density implies lower deployment cost, smaller possibility of traffic jam and radio interference.

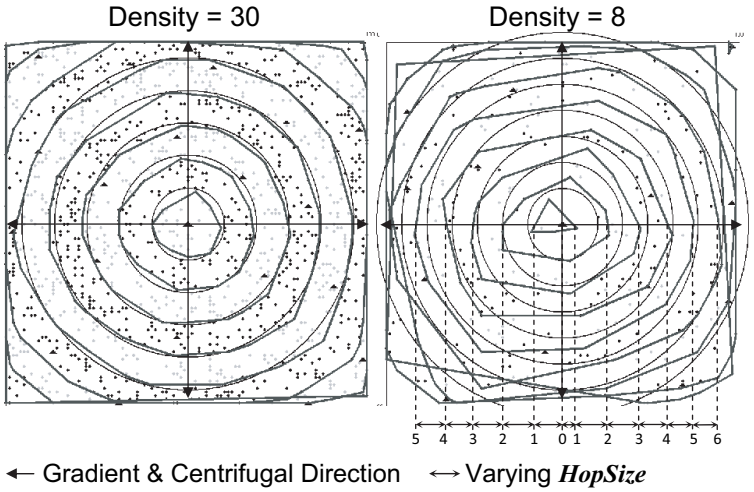


Fig. 3.7: Anisotropy caused by low sensor density.

We argue that the underperformance of Amorphous in sparse networks is caused by network anisotropy. To visualize this anisotropy, we use convex hulls to contain all sensors with the same hop count, which offer a good approximation and illustration

of contour curves in a hop count field. As in the left part of Fig. 3.7, when the sensor density is 30, a tight match can be seen between the concentric rings and the convex hulls, which are represented as the polygons in Fig. 3.7. However, when the sensor density is as low as 8, the convex hulls deviate from concentric rings, with *HopSize* varying unpredictably in different directions. This deviation explains why localization algorithms assuming network isotropy suffer from severe performance degradation in sparse networks.

A CG pattern can be extracted from the hop count fields of sparse networks. In the right part of Fig. 3.7, although the *HopSize* varies when the sensor density is 8, the field gradient still roughly matches the centrifugal direction, which is represented as the rough perpendicularity between centrifugal directions and contour curves. This type of anisotropy is summarized as the Centrifugal Gradient (CG) pattern, which permits varying *HopSize* and preserves a rough match between field gradient and centrifugal direction.

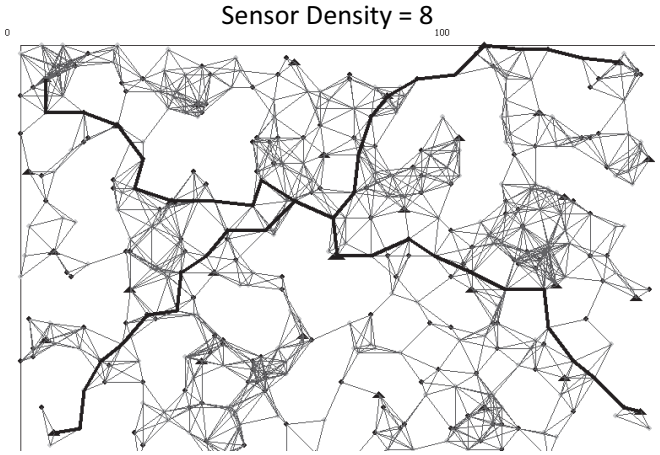


Fig. 3.8: Disturbance by small holes in sparse networks.

The existence of the CG pattern in a sparse network can be explained as its

non-uniform sensor distribution tendency, which creates numerous small holes scattered over the whole network, as illustrated in Fig. 3.8. These small holes distort the shortest path between the anchor and a sensor, which thus slightly deviates from the straight line connecting the two sensor nodes. Therefore, in Fig. 3.7, the *HopSize* varies unpredictably in different directions, but the field gradient is only slightly disturbed from its outward direction, due to the small scale of these holes. This rough match between field gradient and centrifugal direction (the slight detour assumption) is a common trait observable in more generalized network settings, additionally assuming the presence of anisotropic terrains condition, non-uniform sensor distribution, irregular radio propagation and inconsistent sensor radio range.

3.4.2 Proposed DiffTriangle

To tolerate the unpredictable variation of *HopSize* in CG pattern and produce accurate estimates about anchor-sensor distances, we propose DiffTriangle exploiting the rough match between field gradient and centrifugal direction. The DiffTriangle gets its inspiration from the Voronoi diagram with geometrically distributed anchors acting as the sites of the Voronoi cells. Sensors within a Voronoi cell adopts the dominating anchor as their *Reference Station* to revise their distance estimates (to other distant anchors), which deteriorate due to network anisotropy if applying CrMcs or Amorphous.

We assume that the dominating reference station of a Voronoi Cell approximately exhibits the CR pattern to the sensors within the cell. This implies that these reference stations should appear in normal sensors' CR category and thus the distance from sensors to their dominating reference stations should be no more than three

(or four) hops. This assumption implicitly places a demand for anchor distribution density. As an approximate estimation, to guarantee the availability of reference station in the CR category when the sensor density is 10, the anchor percentage ($= \frac{\text{Anchor Number}}{\text{Sensor Number}}$) should be roughly $\frac{One\ Anchor}{(\text{Sensor Density}/\pi r^2)\cdot\pi(3r)^2} = \frac{1}{10}\frac{\pi r^2}{\pi(3r)^2} \approx 1.1\%$. Therefore, in our simulations, a random distribution of anchors with anchor percentage of 3% to 5% can guarantee the availability of reference stations for a large majority of sensors. In those "unavailable" rare cases, CrMcs is used as a backup for DiffTriangle in our prototype system.

The DiffTriangle algorithm gets its abbreviated name from *Differential Triangle*, since frequently the anchor, sensor and reference station (call it *station* for short) are not collinear but are positioned as a triangle depicted by Fig. 3.9. In this triangle,

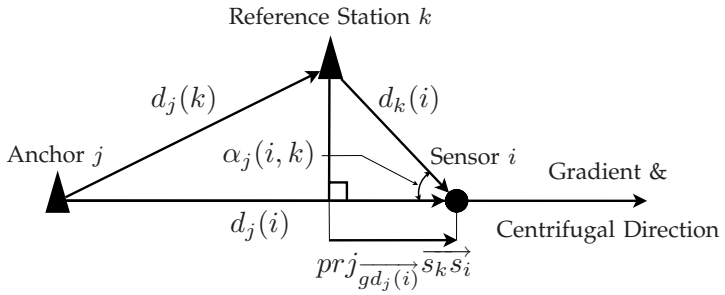


Fig. 3.9: DiffTriangle algorithm in the CG pattern.

we solve the problem of how to revise the estimate of anchor-sensor distance $d_j(i)$, benefiting from:

1. the precisely known $d_j(k)$ from geometric coordinates of anchor j and reference station k;

2. the accurate estimate $\hat{d}_{3k}(i)$ by CrMcs of the station-sensor distance $d_k(i)$, since the nearby reference station k exhibits the CR pattern to sensor i;

3. accurately estimated $prj_{\overrightarrow{gd_j(i)}}\overrightarrow{s_k s_i}$ from the proximity difference of reference station k and sensor i in anchor j's hop count field. Its estimation suffers much less from varying *HopSize* than estimation of $d_j(i)$, thanks to the geometric closeness of reference station k to sensor i.

We adopt the following equations for DiffTriangle Algorithm for the CG pattern.

$$\begin{cases} \hat{d}_{4j}(i)|_k = \sqrt{d_j^2(k) - \hat{d}_{3k}^2(i) + \hat{prj}^2_{\overrightarrow{gd_j(i)}}\overrightarrow{s_k s_i}} + \hat{prj}_{\overrightarrow{gd_j(i)}}\overrightarrow{s_k s_i} \\ \hat{prj}_{\overrightarrow{gd_j(i)}}\overrightarrow{s_k s_i} \quad = \quad [\overline{h}_j(i) - \overline{h}_j(k)] \cdot d_{hop} \end{cases} \tag{3.7}$$

The $\hat{d}_{4j}(i)|_k$ in Eq. (3.7) is our estimate of anchor-sensor distance by DiffTriangle. The station-sensor distance estimate $\hat{d}_{3k}(i)$ required by Eq. (3.7) is obtained by applying CrMcs algorithm in Eq. (3.5). The $\hat{prj}_{\overrightarrow{gd_j(i)}}\overrightarrow{s_k s_i}$ is the proximity difference of sensor i and station k. The adopted proximity $\overline{h}_j(i)$ and $\overline{h}_j(k)$ is the smoothed hop counts of sensor i and station k (see Eq. (3.2)).

Our derivation of the equations of DiffTriangle is presented below. The anchor-sensor distance $d_j(i)$ can be calculated using the following trigonometric formula, assuming the angle $\alpha_j(i,k)$ is known.

$$d_j(i) = \sqrt{d_j^2(k) - d_k^2(i)\sin^2\alpha_j(i,k)} + d_k(i)\cos\alpha_j(i,k) \tag{3.8}$$

The angle $\alpha_j(i,k)$ can be estimated from the projection $prj_{\overrightarrow{s_j s_i}}\overrightarrow{s_k s_i}$ of vector $\overrightarrow{s_k s_i}$ over vector $\overrightarrow{s_j s_i}$, where $\overrightarrow{s_k s_i}$ is the vector pointing from station k to sensor i and $\overrightarrow{s_j s_i}$ is the centrifugal vector from anchor j to sensor i.

$$\alpha_j(i,k) \quad = \quad \arccos\frac{prj_{\overrightarrow{s_j s_i}}\overrightarrow{s_k s_i}}{d_k(i)}$$

The critical $prj_{\overrightarrow{s_j s_i}}\overrightarrow{s_k s_i}$ can be approximated to $prj_{\overrightarrow{gd_j(i)}}\overrightarrow{s_k s_i}$, under the assumption that the centrifugal direction $\overrightarrow{s_j s_i}$ roughly matches the direction of gradient $\overrightarrow{gd_j(i)}$ of hop count field of anchor j seen by sensor i.

$$\alpha_j(i,k) \quad \approx \quad \arccos \frac{prj_{\overrightarrow{gd_j(i)}}\overrightarrow{s_k s_i}}{d_k(i)} \tag{3.9}$$

We estimate the projection $prj_{\overrightarrow{gd_j(i)}}\overrightarrow{s_k s_i}$ in Eq. (3.7) by assuming $prj_{\overrightarrow{gd_j(i)}}\overrightarrow{s_k s_i}$ proportional to the proximity difference $h_j(i) - h_j(k)$ of sensor i and station k, since the isotropy can be well preserved in the small region between sensor i and station k. To achieve higher accuracy, Eq. (3.7) uses smoothed hop count described in Eq. (3.2) for the proximities of sensor i and station k to solve the last hop distance problem, because smoothed hop count is accurate when the hop count is larger than two (see Fig. 3.4).

3.4.3 Error Characteristics of DiffTriangle

In this subsection, we demonstrate by analysis that DiffTriangle can reduce the average distance estimation error to below $0.4r$ in CG pattern, when the sensor density is higher than 8. The analysis is the theoretical foundation of the weighted multilateration adopted by our localization scheme, which configures the expected error of CrMcs as $0.2r$ and that of DiffTriangle (or its enhancement DiffTriangle*) as $0.4r$.

We linearize the impact of several factors on distance estimation error $\Delta d_j(i)$ by applying the Taylor expansion on Eq. (3.8): (1) station-sensor distance estimation error $\Delta d_k(i)$ and (2) angle estimation error $\Delta\alpha_j(i,k)$, which leads to the establishment

of the following equation.

$$\begin{aligned}\Delta d_j(i) \quad \approx \quad & [\cos\alpha_j(i,k) - \frac{d_k(i)\ \sin^2\alpha_j(i,k)}{\sqrt{*}}] \cdot \Delta d_k(i) \\ & + [\sin\alpha_j(i,k) + \frac{d_k(i)\ \sin 2\alpha_j(i,k)}{2\sqrt{*}}]\, d_k(i) \cdot \Delta\alpha_j(i,k) \\ & \text{where } * \ = \ d_j^2(k) - d_k^2(i)\ \sin^2\alpha_j(i,k)\end{aligned}$$

To simplify our analysis, the two items in the form of $\frac{d_k(i)\cdots}{\sqrt{*}}$ are approximated to 0, leading to the simplified representation of $\Delta d_j(i)$ in Eq. (3.10).

$$\Delta d_j(i) \quad \approx \quad \cos\alpha_j(i,k)\Delta d_k(i) + \ \sin\alpha_j(i,k) d_k(i)\Delta\alpha_j(i,k) \tag{3.10}$$

This approximation is reasonable due to the fact that station-sensor distance $d_k(i)$ is much smaller than the anchor-station distance $d_j(k)$. The required $\Delta\alpha_j(i,k)$ in Eq. (3.10) can be derived by applying Taylor Expansion to Eq. (3.9).

$$\Delta\alpha_j(i,k) \quad = \quad \frac{\cos\alpha_j(i,k)\,\Delta d_k(i) - \Delta prj_{\overrightarrow{s_j s_i}}\overrightarrow{s_k s_i}}{d_k(i)\ \sin\alpha_j(i,k)}$$

Therefore, the Eq. (3.10) can be converted to a more simplified form in Eq. (3.11).

$$\Delta d_j(i) \quad \approx \quad 2\ \cos\alpha_j(i,k)\,\Delta d_k(i) - \Delta prj_{\overrightarrow{s_j s_i}}\overrightarrow{s_k s_i} \tag{3.11}$$

This equation indicates that distance estimation error $\Delta d_j(i)$ of DiffTriangle is influenced by two factors: (1) estimation error $\Delta d_k(i)$ of the station-sensor distance; (2) estimation error $\Delta prj_{\overrightarrow{s_j s_i}}\overrightarrow{s_k s_i}$ of the station-sensor proximity difference. The first factor $\Delta d_k(i) < 0.2r$, since the estimation of $d_k(i)$ is achieved using CrMcs, whose accuracy

has been analyzed in the previous section. The second factor $\Delta prj_{\overrightarrow{s_j s_i}}\overrightarrow{s_k s_i} < 0.2r$, since estimation algorithm of $prj_{\overrightarrow{s_j s_i}}\overrightarrow{s_k s_i}$ by Eq. (3.7) minimizes the impact of last hop distance, which is similar to CrMcs. Therefore, according to Eq. (3.11), average distance estimation error $\Delta d_j(i)$ of DiffTriangle in convex anisotropic networks is smaller than $[1 + 2\cos\alpha_j(i,k)] \cdot 0.2r \approx 0.4r$, which is consistent with the simulation results in Section 3.7.

3.5 DG Pattern and Proposed DiffTriangle*

This section focuses on the distance estimation problem in anisotropic networks additionally assuming the presence of large obstacles. These obstacles can distort field gradients to strongly deviate from the centrifugal directions (see region III in Fig. 3.1 and Fig. 3.10), which undermines the slight detour assumption of DiffTriangle (the rough match between gradient direction and centrifugal direction) and deteriorates its accuracy. In this section, we firstly quantify the impact of the DG pattern on DiffTriangle and based on the analysis propose DiffTriangle* - an enhancement to DiffTriangle, which can tolerate both obstacle detour and the interference of multiple anisotropic factors.

Our DiffTriangle* has the following advantages compared with other algorithms tolerating detoured anchors. Different from RenderedPath [Li and Liu, 2007] assuming a constant number of anchors, DiffTriangle* can provide higher accuracy, due to its ability inherited from DiffTriangle to exploit increased anchor density and tolerate multiple anisotropic factors. Compared with PDM [Lim and C., 2005], which is another algorithm able to tolerate both multiple anisotropic factors and obstacle detour, DiffTriangle* is a distributed solution with less communication overhead and

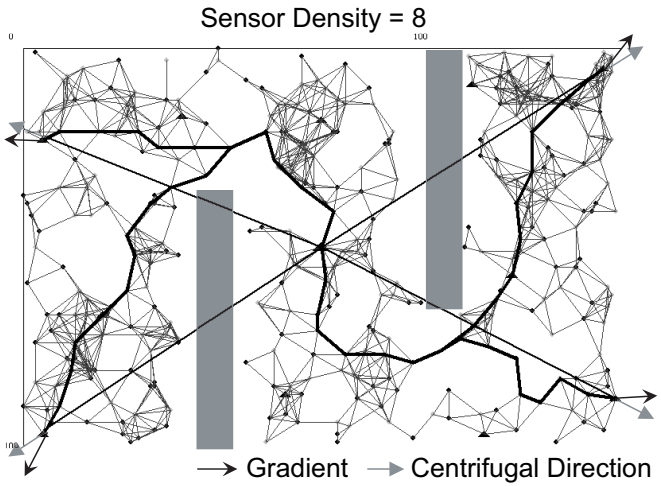


Fig. 3.10: The deviation of gradient and centrifugal direction in the DG pattern.

is robust in all simulated networks (rectangular, U-shaped, O-shaped). PDM however degrades severely in the O-shaped deployment fields (see Subsection 3.7.5). In contrast to GridVoting [Liu et al., 2005], which can detect and drop outliers including the detoured anchors, DiffTriangle* has higher accuracy, since GridVoting can filter strongly detoured anchors but not moderately detoured anchors, which thus deteriorates its localization accuracy. Moreover, GridVoting is sensitive to the configured threshold of distance estimation error.

3.5.1 DG pattern and Its Impact on Accuracy of DiffTriangle

In this subsection, we analyze the impact of the DG pattern on anchor-sensor distance estimation accuracy of DiffTriangle, showing the accuracy of DiffTriangle can degrade to $0.8r$ in typical anisotropic concave networks. This error may not be tolerated by many location dependent protocols and applications, since $0.4r$ is the tolerable bound of localization error as argued by [He et al., 2003]. This imposes a requirement for more effective anchor-sensor distance estimation algorithms to handle

obstacle detour, which is the topic of the next subsection.

The shortest path between two sensor nodes can strongly deviate from the straight line connecting them, if distorted by obstacles, as illustrated by the exemplified S-shaped network in Fig. 3.10. This deviation causes a mismatch between field gradient and centrifugal direction, which we summarize as the DG pattern, which undermines the assumption of DiffTriangle and degrades its accuracy. We quantify the degree of obstacle detour by the deviation angle $\gamma_j(i)$.

$$\gamma_j(i) \quad : \quad \text{the deviation angle between gradient direction } \overrightarrow{gd_j(i)} \text{ and centrifugal direction } \overrightarrow{s_j s_i}$$

Based on this definition, we can quantify the impact of obstacle detour on DiffTriangle's accuracy as

$$\pm \sin \alpha_j(i,k) \cdot d_k(i) \cdot \gamma_j(i). \tag{3.12}$$

The above quantification is based on the ambiguous DiffTriangle in Fig. 3.11, which clearly shows the deviation angle $\gamma_j(i)$. Due to the existence of nonzero $\gamma_j(i)$, sensor

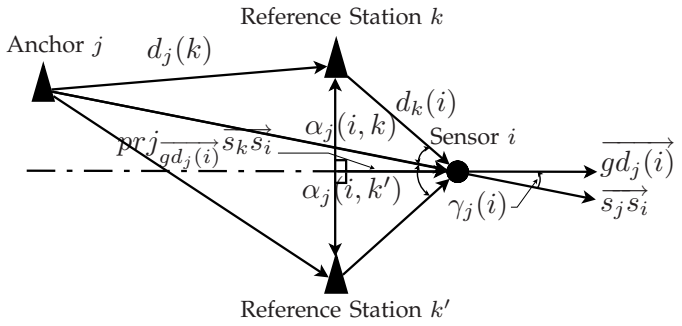


Fig. 3.11: Ambiguous DiffTriangle in the DG pattern.

i can not tell whether its reference station and the anchor lie on the same side of the gradient $\overrightarrow{gd_j(i)}$ or on opposite sides. This ambiguity is represented in Fig. 3.11 as the two cases, reference stations k and k'. Caused by this ambiguity of stations k and k', the $\alpha_j(i,k)$ required by DiffTriangle in its Fig. 3.8 has ambiguous values as represented by $\alpha_j(i,k)$ and $\alpha_j(i,k')$ in Fig. 3.11. Therefore, Eq. (3.9) for DiffTriangle should be modified to the following equation for ambiguous DiffTriangle.

$$\alpha_j(i,k),\ \alpha_j(i,k') \ \approx\ \arccos\left[prj_{\overrightarrow{gd_j(i)}}\overrightarrow{s_k s_i}\ /\ d_k(i)\right] \pm \gamma_j(i)$$

Therefore, the impact of the ambiguous deviation angle $\pm\gamma_j(i)$ on accuracy of DiffTriangle can be quantified as $\pm \sin\alpha_j(i,k) \cdot d_k(i) \cdot \gamma_j(i)$, by assigning $\pm\gamma_j(i)$ to $\Delta\alpha_j(i,k)$ in Eq. (3.10).

Our conclusion is that the accuracy of DiffTriangle in the DG pattern can degrade to $0.4r + 0.4r = 0.8r$, since the accuracy of DiffTriangle in the CG pattern is roughly $0.4r$ and the additional impact of obstacle detour on the accuracy of DiffTriangle is $\pm 0.4r$. This impact $\pm 0.4r$ of obstacle detour is estimated by Eq. (3.12) as $\pm\frac{1}{2} \cdot 1.6r \cdot \frac{\pi}{6} \approx \pm 0.4r$, since in typical concave anisotropic networks, the average station-sensor distance $d_k(i)$ is roughly 2 hops $\approx 1.6r$ and the average deviation angle $\gamma_j(i)$ is $\frac{\pi}{6}$ (see Fig. 3.10). Moreover, the impact $\pm 0.4r$ of obstacle detour indicates that distance estimates by DiffTriangle may be either enlarged or diminished.

3.5.2 Proposed DiffTriangle* and wMultilateration(8)

To tolerate the gradient distortion (or obstacle detour), we present our solution comprising of two parts: *wMultilateration(n)* and *DiffTriangle**. wMultilateration(8) is a weighted multilateration [Foy, 1976] method using only the nearest 8 anchors.

- wMultilateration(8) deploys the *Nearest(8)* filter at the CG recognizer to reduce the chance to contain detoured anchors in the CG category, assuming the nearest 8 anchors are less vulnerable to obstacle detour than farther anchors. We choose the threshold eight, because eight is beneficial to mitigate the potential bad geometry effect of anchors [He et al., 2003].

- wMultilateration(8) uses weighted multilateration [Foy, 1976] to prefer nearby anchors, assuming the expected error of CR pattern is $0.2r$ and that of the CG pattern is $0.4r$.

However, there is a non-negligible chance for the CG category to contain detoured anchors, which degrades the performance of DiffTriangle and deteriorates the localization accuracy.

We therefore propose DiffTriangle*, an enhancement to DiffTriangle algorithm with no additional communication cost and able to generate accurate distance estimates with the presence of gradient distortion (call it "recover the distorted anchors" for short). Therefore, our framework in Fig. 3.2 adopts DiffTriangle* as a better distance estimator for the CG pattern than DiffTriangle, since it is inevitable for the CG category to contain detoured anchors. Although DiffTriangle* has higher accuracy than DiffTriangle, it needs two reference stations from the CR category. Therefore, when there is only one anchor in the CR category, DiffTriangle is a good backup for DiffTriangle*; when there is none, CrMcs is the last choice.

We present the intuition of DiffTriangle* as follows. DiffTriangle* assumes there are two reference stations for sensor i: *Station* k and *altStation* k^* with $d_k(i) < d_{k^*}(i)$. Its main idea is that if sensor i and station k both use the DiffTriangle algorithm and altStation k^* (as the reference station required by DiffTriangle) to

estimate their distances to anchor j, then the two distance estimates $\hat{d}_{4j}(i)|_{k^*}$ and $\hat{d}_{4j}(k)|_{k^*}$ by DiffTriangle may encounter similar distortion effects and thus have similar estimation errors. Because the error in $\hat{d}_{4j}(k)|_{k^*}$ is known from the geometric location of station k, it is possible for DiffTriangle* to recover the error in $\hat{d}_{4j}(i)|_{k^*}$ due to obstacle detour by knowing the error in $\hat{d}_{4j}(k)|_{k^*}$. As a summary, DiffTriangle* is a revision to DiffTriangle (and thus a second order revision to CrMcs), exploiting the geometric closeness between sensor j and station k and recovering the bias in DiffTriangle's distance estimation due to obstacle detour.

We present the equation of DiffTriangle* as follows.

$$\begin{aligned} \hat{d}_{5j}(i)|_{k^*}^{k} &= \hat{d}_{4j}(i)|_{k^*} - \Delta\hat{d}_{4j}(i)|_{k^*} \\ &= \hat{d}_{4j}(i)|_{k^*} - \Delta\hat{d}_{4j}(k)|_{k^*} \cdot \frac{\hat{d}_{3k^*}(i)}{\hat{d}_{3k^*}(k)} \end{aligned} \tag{3.13}$$

In the above equation, the sensor i firstly obtains an unrevised anchor-sensor distance estimate $\hat{d}_{4j}(i)|_{k^*}$ by DiffTriangle algorithm (with altStation k^* as the required reference station). Then we revise the $\hat{d}_{4j}(i)|_{k^*}$ to $\hat{d}_{5j}(i)|_{k^*}^{k}$ by the estimated error $\Delta\hat{d}_{4j}(i)|_{k^*}$ in $\hat{d}_{4j}(i)|_{k^*}$. The estimated error $\Delta\hat{d}_{4j}(i)|_{k^*}$ is a linear transformation from the estimated error $\Delta\hat{d}_{4j}(k)|_{k^*}$ in the anchor-station distance estimation. The above correction is possible, since the station k and the sensor i are geometrically close to each other and the anchor j's hop field near them thus probably experiences similar distortion with similar deviation angle $\gamma_j(i) \approx \gamma_j(k)$. The aim in Eq. (3.13) of multiplying the revision $\Delta d_{4j}(k)$ by the ratio $\frac{\hat{d}_{3k^*}(i)}{\hat{d}_{3k^*}(k)}$ is to remove the impact of sensor-altStation distance $d_{k^*}(i)$, since our analysis indicates that $d_{k^*}(i)$ linearly interferes the accuracy of DiffTriangle.

We present the analysis on error characteristics of DiffTriangle, showing that $d_{k^*}(i)$ linearly interferes the accuracy of DiffTriangle. According to Eq (3.10), if sensor i applies DiffTriangle and uses altStation k^* as the reference station, the distance estimation error of DiffTriangle can be estimated by the following equation.

$$\Delta\hat{d}_{4j}(i)|_{k^*} \quad \approx \quad \cos\alpha_j(i,k^*)\,\Delta d_{k^*}(i) + \sin\alpha_j(i,k^*)\,d_{k^*}(i)\,\Delta\alpha_j(i,k^*)$$

From the above equation, we know the accuracy of DiffTriangle is mainly affected by two factors: the altStation-sensor distance $d_{k^*}(i)$ and the deviation angle $\gamma_j(i)$ from obstacle detour. According to our analysis in the previous subsection, the deviation angle $\gamma_j(i)$ directly interferes our estimation of angle $\alpha_j(i,k^*)$, whose impact over $\Delta d_{4j}(i)$ is $\pm\sin\alpha_j(i,k^*)\cdot d_{k^*}(i)\cdot\gamma_j(i)$. The aim of DiffTriangle* is to recover this impact from obstacle detour. The other factor affecting accuracy of DiffTriangle is $d_{k^*}(i)$, because (1) there exists a tight bond (linear approximately) between $d_{k^*}(i)$ and the altStation-sensor distance estimation accuracy $\Delta d_{k^*}(i)$ due to the existence of error accumulation if applying the isotropic CrMcs method; (2) $d_{k^*}(i)$ linearly amplifies the error $\Delta\alpha_j(i,k^*)$ in estimation of angle $\alpha_j(i,k^*)$.

The required error $\Delta\hat{d}_{4j}(k)|_{k^*}$ (or correction) in Eq. (3.13) is provided by station k as follows.

$$\Delta\hat{d}_{4j}(k)|_{k^*} \quad = \quad \hat{d}_{4j}(k)|_{k^*} - d_j(k).$$

In this equation, the station k derives an anchor-station distance estimate $\hat{d}_{4j}(k)|_{k^*}$, by pretending as a normal sensor and applying the DiffTriangle algorithm with altStation k^* as the required reference station. The error in this estimate $\hat{d}_{4j}(k)|_{k^*}$ can be known,

since the anchor-station distance $d_j(k)$ can be known from their coordinates.

An implementation tip of DiffTriangle* for reduced communication overhead is that the required correction $\hat{\Delta}d_{4j}(k)$ in Eq. (3.13) provided by the station k can be calculated locally by sensor i, since the station k has already transmitted its Input I to sensor i and the Input II of the station k (that contains the Input I of altStation k^*) is the same with Input II of sensor i (see Fig. 3.2). Another implementation tip of DiffTriangle* for an improved accuracy is to well handle the case that the station k and altstation k^* are roughly of the equal distance to the sensor i. To well handle this, sensor i can make anchors k and k^* take turns to be the station and the altStation. Then sensor i by applying Eq. (3.13) can obtain two revised distance estimates to anchor j: $\hat{d}_{5j}(i)|_{k^*}^{k}$ and $\hat{d}_{5j}(i)|_{k}^{k^*}$. The final estimate $\hat{d}'_{5j}(i)$ is a weighted average of the two estimates.

$$\hat{d}'_{5j}(i)|_{k^*}^{k} = \frac{1}{\hat{d}_{3k}(i) + \hat{d}_{3k^*}(i)} [\quad \hat{d}_{3k^*} \cdot \hat{d}_{5j}(i)|_{k^*}^{k} + \hat{d}_{3k} \cdot \hat{d}_{5j}(i)|_{k}^{k^*}] \tag{3.14}$$

3.6 Pattern-driven Localization Algorithm

We present the pseudo code of our pattern-driven localization algorithm in Alg. 1, which corresponds to the framework depicted in Fig. 3.2. The pseudo code in Alg. 1 integrates (provides a facade for) the three proposed algorithms:

- CrMcs at line 5 for the isotropic anchors in the CR category to mitigate the impact of last hop distance,

- DiffTriangle at line 10 requiring one reference station (for the first order revision to CrMcs) to tolerate the anisotropic anchors in the CG category,

- DiffTriangle* at line 8 requiring two reference stations (for the second order revision to CrMcs) to tolerate the detoured and anisotropic anchors in the CG category.

An implementation tip for the weighted multilateration [Foy, 1976] at line 13 is that it is necessary to use multiple start points to perform the multilateration and choose the best solution (with the smallest weighted average residue) as the final location estimate, due to its non-negligible possibility of trap in local minima. One rule of thumb for the reference station filtering (and propagation) at line 6 is that (1) when the anchors are densely distributed, we only need reference stations within three hops; (2) when the anchors are sparsely distributed, the threshold can be relaxed to be four hops. For the meaning of used annotations, please refer to Fig. 3.2 and TABLE 3.1.

3.7 Simulation Results

To verify the effectiveness of our pattern-driven localization scheme in tolerating multiple network anisotropic factors, simulations have been designed and conducted in this section. In these simulations, the accuracy of our scheme has been compared with that of Amorphous [Nagpal et al., 2003] and PDM [Lim and C., 2005] in various network configurations. We choose Amorphous and PDM for comparisons, since (1) Amorphous is a typical isotropic localization algorithm which can be used by an anisotropy-tolerating algorithm to show its power and (2) PDM is a state-of-the-art algorithm who also declares to handle network anisotropy. We do not compare our algorithm with iMultihop [Wang and Xiao, 2006], REP [Li and Liu, 2007] or MDS-MAP(P) [Shang and Ruml, 2004], since they are not the algorithms declared to

tolerate multiple anisotropic factors. Our conclusions are (1) compared with Amorphous, PDM and our scheme can improve both distance estimation accuracy and localization accuracy by tolerating multiple anisotropic factors; (2) compared with PDM, our scheme has lower communication cost and is more accurate in dense networks; (3) our scheme is robust in rectangular, O-shaped, U-shaped networks, while PDM degrades severely in O-shaped networks.

3.7.1 Evaluation Metrics and Controlled System Parameters

We use the following metrics to evaluate the three algorithms (Amorphous, PDM and our scheme) in simulations.

- Average distance estimation error μ_h for h-hop paths:

$$\mu_h = \sum_{\hat{d}_j(i) \in \mathcal{D}_h} \frac{|\hat{d}_j(i) - d_j(i)|}{|\mathcal{D}_h|}, \text{ where}$$
$$\mathcal{D}_h = \{\text{distance estimates of all paths with lengths of } h\}.$$

  μ_h is a useful metric to observe the accumulation of distance estimation error with the increase of hop count.

- Overall average distance estimation error μ: $\mu = \sum_{\hat{d}_j(i) \in \cup\mathcal{D}_h} \frac{|\hat{d}_j(i) - d_j(i)|}{|\cup\mathcal{D}_h|}$, where $\cup\mathcal{D}_h$ is the set of distance estimates of paths with any hops. According to the analysis of multilateration in [Savvides et al., 2003a] based on Cramér-Rao bound, localization error should approach average distance error μ, which we shall verify in our simulations.

- Average localization error ε: ε is the arithmetic mean of the localization error

of all sensors, with $\varepsilon = \sum \varepsilon_i / N$, where N is the total number of sensors and ε_i is the localization error of sensor i with $\varepsilon_i = |\hat{p_i} - p_i|$, which is the geometric distance between the estimated position $\hat{p_i}$ and the true position p_i.

We control the following system parameters and investigate their impact on the above evaluation metrics.

- Sensor Density (SD): The average number of sensors per sensor radio area. SD $= \overline{|N(i)|}$. (see TABLE 3.1).

- Degree of Radio Irregularity (DOI): With the presence of DOI, the possibility $P(d)$ of two sensor nodes with distance d to establish a link follows the equation below.

$$P(d) = \begin{cases} 1 & \frac{d}{r} < 1 - \text{DOI} \\ \frac{1}{2\text{DOI}} \left(\frac{d}{r} - 1\right) + \frac{1}{2} & 1 - \text{DOI} \leq \frac{d}{r} \leq 1 + \text{DOI} \\ 0 & \frac{d}{r} > 1 + \text{DOI} \end{cases}$$

- Shape of Deployment Region (SDR): The rectangular, U-shaped, O-shaped regions have been studied.

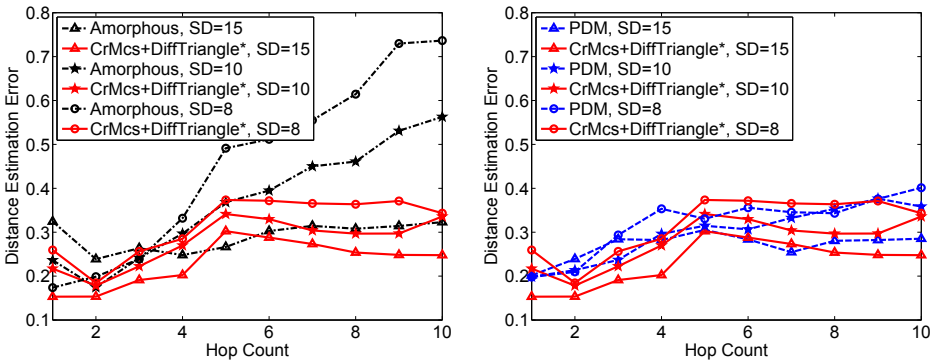
- Anchor Cell Radius (ACR): The average radius of the Voronoi cell dominated by an anchor. ACR $(= \sqrt{\frac{\text{Area of deployment region}}{\pi \cdot \text{Anchor Number}}})$ is a good indicator for the average distance of a sensor to its nearest reference station.

3.7.2 Distance Estimation Error when Varying Sensor Density

This simulation varies the sensor density to study its impact on the accuracy of distance estimation. To isolate the impact of sensor density, we intentionally fix the DOI to 0.0, the ACR to $1.5r$ and the SDR to rectangular regions ($10r \times 10r$). The simulation results are plotted in Fig. 3.12.

We show the advantages of our scheme over Amorphous in Fig. 3.12(a). (1) When SD is as low as 8, Amorphous encounters severe error accumulation as the hop count grows, due to the anisotropy from low sensor density (Subsection 3.4.1). However, Amorphous's accuracy is relatively satisfactory in the first three (or four) hops. That is why our scheme approximately treats the first few hops as isotropic regions, where the impact of varying *HopSize* is negligible and the CR pattern approximately applies. Compared with Amorphous, our scheme can suppress the error accumulation and keep the distance estimation error constantly below $0.4r$, which is consistent with our analysis of DiffTriangle's accuracy in Subsection 3.4.3. (2) Even in the approximately isotropic region within four hops, our scheme constantly outperforms Amorphous, because CrMcs is a better method to solve the last hop distance problem than Amorphous smoothing.

We compare our scheme with PDM in Fig. 3.12(b). (1) In dense networks (SD $\geq$ 15), the accuracy of PDM is still above $0.2r$, since it neglects the last hop distance problem. As a comparison, the accuracy of our scheme can be improved to be around $0.15r$, when sensor density is 15 and in the first four hops. However, our scheme is only slightly better than PDM when sensor density is 15 and beyond four hops, since the anisotropy (inaccurate $HopSize$) becomes the dominating factor rather the last hop



(a) Amorphous and Our scheme. (b) PDM and Our scheme.

Fig. 3.12: Distance estimation error when varying SD, when DOI = 0.0, Anchor Number = 6, SDR = rectangle $10r \times 10r$, ACR = $1.5r$.

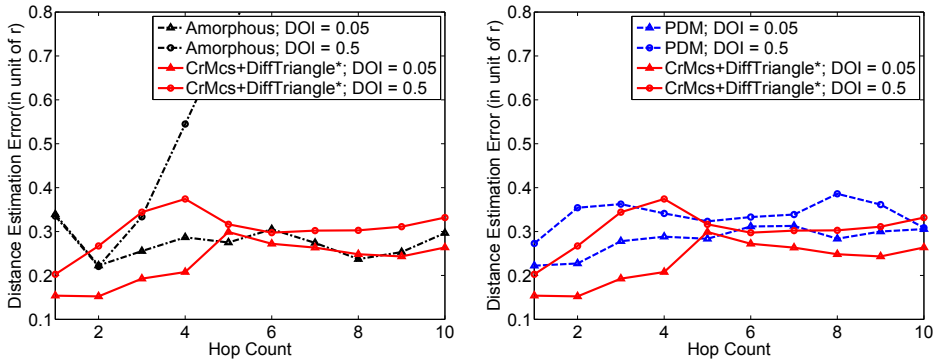
distance. (2) In sparse networks (SD $\leq$ 10), PDM has non-negligible possibility for the failure of its matrix inverse operation and may have to apply the pseudo inverse multiple times. Moreover, sparse networks have higher possibility to be separated and the branch separated from the base station may not be localized by PDM. Our scheme, a distributed algorithm, has no such problem.

3.7.3 Distance Estimation Error when Varying DOI Ratio

This simulation investigates the impact of high DOI ratio, such as DOI = 0.5, on distance estimation accuracy. To highlight its impact, we minimize the impact of other anisotropic factors by configuring SD as high as 15 and SDR as rectangle regions. The simulation results are depicted in Fig. 3.13.

In Fig. 3.13(a), the accuracy of Amorphous degrades severely when DOI = 0.5, since the Kleinrock's equation that Amorphous uses is inaccurate in estimating *Hop-Size*, assuming a perfect circular communication range. As a comparison, the performance of our scheme however remains stable, when DOI = 0.5, although our scheme

also uses the Kleinrock's equation. This is because the estimated *HopSize* is only used by DiffTriangle to estimate the short sensor-station distance and the projection of this distance on the field gradient. As a summary, our DiffTriangle and DiffTriangle* method is insensitive to the error in *HopSize* estimation, which gives our scheme the freedom to choose any appropriate *HopSize* estimation algorithm, either online or offline. In Fig. 3.13(b), the PDM is also robust against radio irregularity. The slightly lower accuracy of PDM than our scheme is also due to the non-negligible last hop distance problem in dense networks.

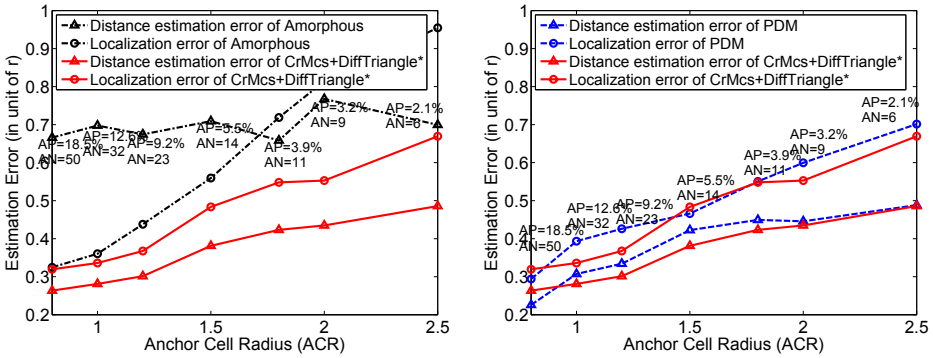


(a) Amorphous and Our scheme. (b) PDM and Our scheme.

Fig. 3.13: Distance estimation error varying DOI ratio; SD = 15, SDR = rectangle $10r \times 10r$, ACR = $1.5r$.

3.7.4 Localization Error when Varying ACR

This simulation explores the impact of anchor density (quantified by ACR) over the accuracy of distance estimation and localization. The simulation results are illustrated by Fig. 3.14, in which we adopt the same localization algorithms wMultilateration(8) for Amorphous, PDM and our scheme to maintain a fair comparison.



(a) Amorphous and Our scheme. (b) PDM and Our scheme.

Fig. 3.14: Distance estimation error and localization error varying ACR; SD = 8, DOI = 0.25, SDR = rectangle $10r \times 10r$, Sensor Number $\approx$ 250, Anchor Percentage (AP) = $\frac{\text{Anchor Number (AN)}}{\text{Sensor Number}}$.

In Fig. 3.14(a), the distance estimation accuracy of Amorphous has no change when ACR decreases, since Amorphous has no mechanism exploiting the increased anchor density and optimizing its distance estimation accuracy. In contrast, our scheme and PDM have much smaller distance estimation error below $0.4r$. Although the distance estimation accuracy of Amorphous does not improve with reduced ACR, its localization accuracy constantly improves, if Amorphous adopts wMultilateration(8) for localization. However, Amorphous's localization accuracy deterioration speed (when ACR grows) is much faster than our scheme and PDM. Moreover, for PDM and our scheme, we can find a tight correspondence between average distance estimation error and average localization error, which is consistent with the theoretical analysis in [Savvides et al., 2003a] based on Cramér-Rao lower bound.

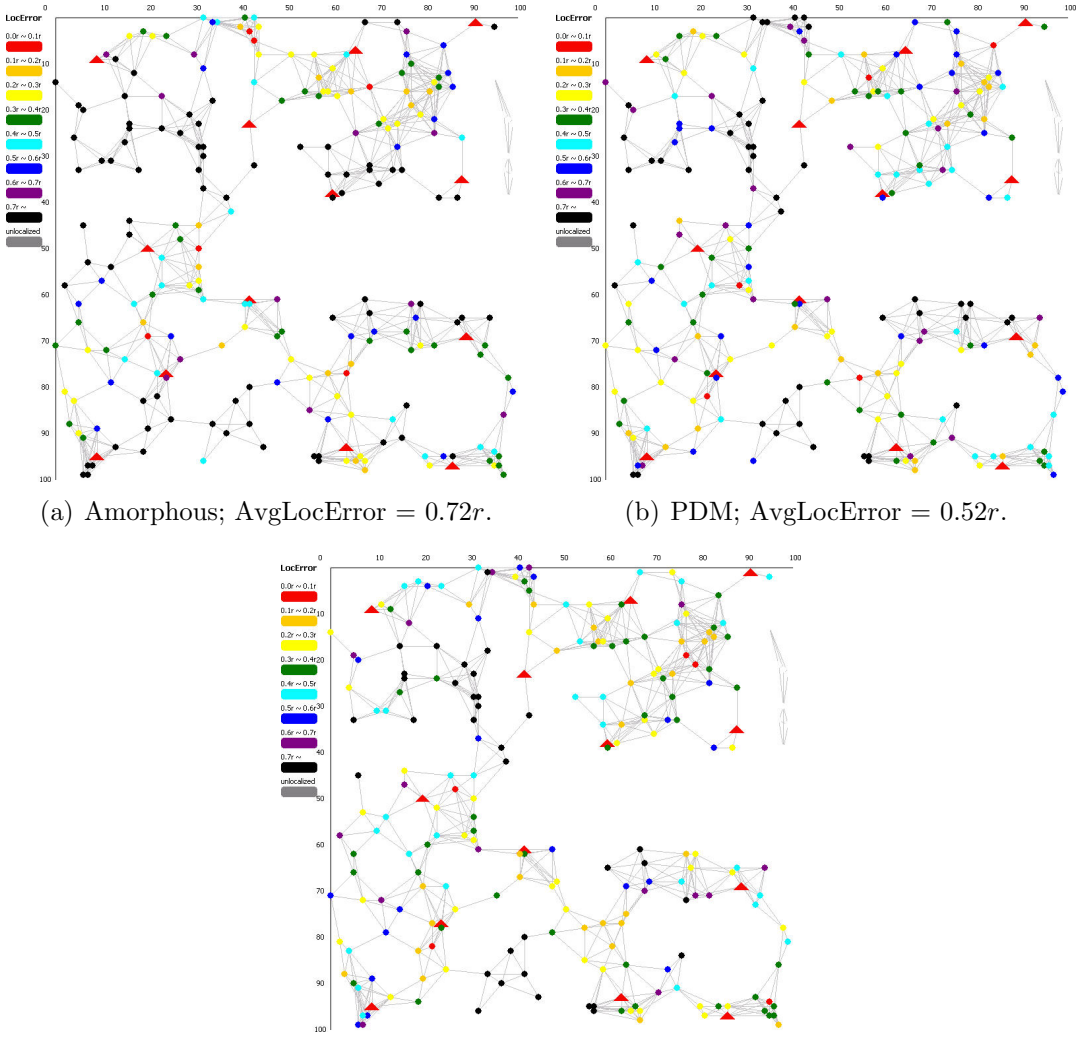
The underperformance of Amorphous is because although wMultilateration(8) uses the nearest 8 anchors and the weighted multilateration to prefer nearby anchors, it can be quite difficult to precisely capture the error accumulation trend of Amorphous's distance estimation at the location estimator layer. As a comparison, PDM

and our scheme choose to suppress the error accumulation mainly at the distance estimator layer by exploiting the dense anchor distribution. For this reason, the localization accuracy of Amorphous (even with wMultilateration(8)) can be above $0.7r$, while that of PDM and our scheme stay below $0.5r$ when ACR is below 2. This performance improvement is important, since the work in [He et al., 2003] has stated that the performance of several location dependent protocols (like geometric routing) degrades quickly when the localization accuracy is above $0.4r$. Moreover, when anchors are densely distributed with ACR $= 1.5r$, the anchor percentage is merely 5.5%, which does not seem to be an unbearable huge investment. The optimized tradeoff between anchor percentage and localization accuracy may only be found in specific applications.

3.7.5 Localization Error when Varying SDR

In this subsection, we compare the localization accuracy of three algorithms (i.e. Amorphous, PDM and our scheme) in anisotropic networks with both obstacle detour and the presence of multiple anisotropic factors (radio irregularity and low sensor density). This simulation shows that (1) in the U-shaped regions like Fig. 3.15, PDM and our scheme can improve the localization accuracy, compared with Amorphous; (2) in the O-shaped regions like Fig. 3.16, PDM degrades dramatically but our algorithm still functions well. In Fig. 3.15&3.16, we depict the network topologies as graphs with triangular nodes for anchors, circular nodes for sensors, links for radio connections. We also label for each node its localization error by color.

In Fig. 3.15, we compare the localization accuracy of Amorphous, PDM and our scheme in the U-shaped region (for fairness, all of them use wMultilateration(8) for



(a) Amorphous; AvgLocError = $0.72r$. (b) PDM; AvgLocError = $0.52r$.

(c) Our scheme; AvgLocError = $0.46r$.

Fig. 3.15: A comparison of localization accuracy of Amorphous, PDM and our scheme; SD = 8, DOI = 0.3, ACR = $1.6r$, r = 10ft.

location estimation). As illustrated, Amorphous suffers from network anisotropy and obstacle detour, with localization accuracy above $0.7r$. If instead applying PDM for anchor-sensor distance estimation in Fig. 3.15(b), the localization accuracy can be improved to around $0.5r$. When we use our scheme (including CrMcs and DiffTriangle*) to estimate the distances in Fig. 3.15(c), we can achieve similar localization accuracy with PDM. Our scheme however has two advantages over PDM.
(1) To achieve this comparable accuracy with PDM, our scheme has simpler communication operation (i.e., the confined flooding) and less communication overhead.

- The overall communication overhead of PDM is $O(M^2N)$, where M is the number of anchors and N is the number of sensors. PDM firstly needs the M times flooding initiated by M anchors to make the sensors know their hop counts, whose overhead is $O(MN)$. PDM then needs to collect the hop counts between all pair of anchors to the base station (i.e. M unicasts to the base station), and disseminate the calculated PDM matrix to the entire network (i.e. a flooding of a $M \times M$ matrix), whose communication cost is $O(M^2N)$.

- The overall communication overhead of our algorithm is $O(MN)$. (1) Besides the M times flooding initiated by M anchors, our algorithm (precisely DiffTriangle and DiffTriangle*) requires each anchor to propagate its hop counts (to M anchors) to three or four hops' neighborhood by a confined flooding. This is equivalent to $1 \sim 2$ times flooding of a M-sized vector because a confined flooding may partly overlap with another confined flooding. This has the overhead of $O(MN)$. (2) Another kind of communication needed by our algorithm (precisely CrMcs) is to make every node aware of the number of neighbors with equal and larger hop counts. This needs N times local broadcast of M sized

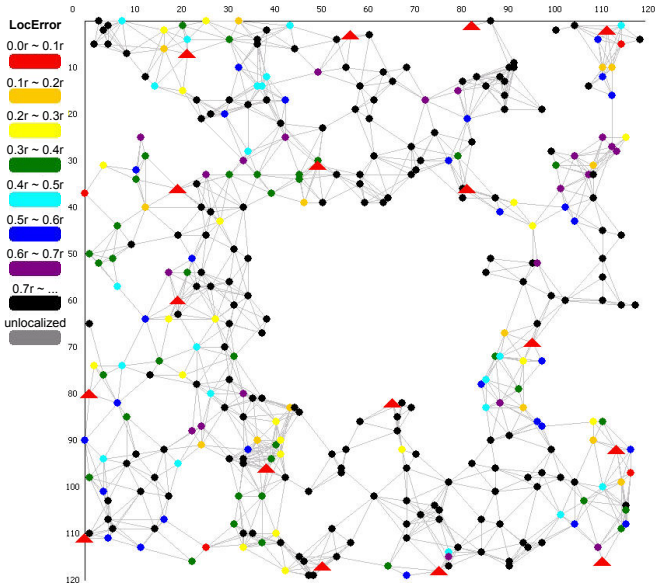
vector (see paragraph 3 of Section 3.2), whose overhead is $O(MN)$.

(2) Our scheme has specially optimized localization accuracy near each anchor (see Fig. 3.15(c)), since sensors use their nearest reference stations to optimize distance estimates. The shorter the distance to reference station, the better the localization accuracy.

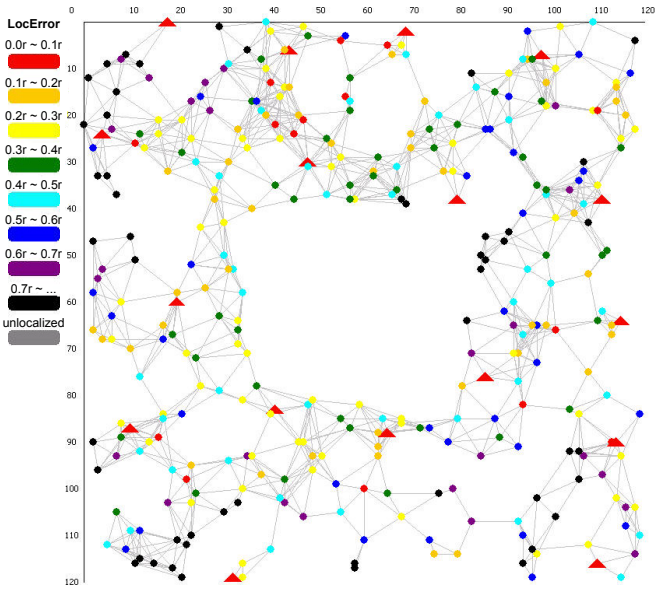
Here comes an interesting feature of our scheme in contrast to PDM: PDM degrades in the O-shaped region, but our scheme still functions well as shown by Fig. 3.16. According to the simulation in Fig. 3.16(a), the localization accuracy of PDM degrades quickly in the O-shaped region. This degradation is probably because the PDM tries to find an optimum linear transformation between two high dimensional spaces (One space is the proximity distances to all anchors and the other is the Euclidean distances to all anchors). This bidirectional linear transformation provides a good approximation in the rectangular, U-shaped and S-shaped regions, but it is inaccurate in the O-shaped regions. Our localization scheme has no such problem and it still renders satisfying performance in the O-shaped region as illustrated by Fig. 3.16(b).

3.8 Conclusion

For accurate localization in networks with multiple anisotropic factors, we propose a pattern-driven localization scheme, applying different distance estimation algorithms for anchors exhibiting different patterns, i.e., the CR, CG and DG patterns. For the CR pattern, CrMcs is adopted to minimize the impact of last hop distance. For the CG pattern, DiffTriangle is used to tolerate varying *HopSize* and exploiting



(a) PDM; AvgLocError = $0.87r$.



(b) Our scheme; AvgLocError = $0.45r$.

Fig. 3.16: A comparison of localization accuracy of PDM and our scheme in O-shaped region; SD = 8, DOI = 0.3, ACR = $1.6r$, r = 10ft.

the rough match between field gradient and centrifugal direction. DiffTriangle* provides an enhancement to DiffTriangle to additionally tolerate obstacle detour. For the DG pattern, where the line-of-sight rule no longer holds, the anchors are dropped. Both theoretical analysis and simulation results support the effectiveness of our localization scheme in tolerating multiple network anisotropic factors.

Table 3.1: Symbols used by Our Pattern-Driven Range-Free Localization Scheme

	Networks
r	average communication range of sensors and anchors
loc_i	the location estimate of sensor i. If sensor i is an anchor, then this knowledge is precise; otherwise, it is imprecise.
$d_j(i)$	the real geometric distance between sensor i and anchor j
$h_j(i)$	the hop count of sensor i with respect to anchor j
$N(i)$	the set of immediate neighbors of sensor i, including i itself
SD	Sensor Density = $\overline{\lvert N(i)\rvert}$, i.e., the expected number of immediate neighbors of a sensor
	DV-Hop
$\hat{d}_{1j}(i)$	an estimate of $d_j(i)$ given by DV-Hop
d_{hop}	average distance per hop or *HopSize*
	Amorphous
$\hat{d}_{2j}(i)$	an estimate of $d_j(i)$ given by Amorphous
$\overline{h}_j(i)$	the smoothed hop count by Amorphous
	CrMcs
$\hat{d}_{3j}(i)$	an estimate of $d_j(i)$ given by CrMcs
$r(h)$	the radius of h^{th} hop communication disks
$dsk_i(h)$	h^{th} hop communication disk of sensor i; $dsk_i(h)$ centers at sensor i, with radius $r(h)$
$a_j(i)$	the intersected area of $dsk_i(1)$ and $dsk_j(h_j(i))$
$N_j(i)$	$\{\, l \mid l \in N(i) \ \wedge \ h_j(l) \leq h_j(i) \,\}$
	DiffTriangle
$\hat{d}_{4j}(i)\vert_k$	an estimate of $d_j(i)$ revised by reference station k
$\overrightarrow{gd_j(i)}$	the gradient of anchor j's hop count field at sensor i
$\overrightarrow{s_i s_j}$	the vector from sensor i to sensor j
	DiffTriangle*
$\hat{d}_{5j}(i)\vert_{k^*}^{k}$	an estimate of $d_j(i)$ revised by reference stations k and k^*
$\gamma_j(i)$	the deviation angle between $\overrightarrow{gd_j(i)}$ and $\overrightarrow{s_j s_i}$

Algorithm 1: Our localization scheme running on sensor i

Input:
1. DGPattern (Input I): the initial set of heard anchors (by flooding and local broadcast), temporally stored as DGPattern = Input I = { [j, loc_j, $h_j(i)$, $|N_j(i)|$, $|N(i)|$] }
2. RefInfo (Input II): the Input I of nearby reference stations propagated by confined flooding, with RefInfo = { [k, Input I] }, where k is the reference station id. RefInfo.size indicates the number of heard stations.
Output: location estimate loc_i of sensor i.

```
1  begin
       /* Init:  System Assembly                                          */
2      CRThreshold = 4
3      CRRecognizer = HopNoLargerThan(CRThreshold)
4      CGRecognizer = Nearest(8)
5      CREstimator = CrMcs                          /* Eq. (3.5)  */
6      RefInfo = HopNoLargerThan(CRThreshold).filter(RefInfo)
7      if RefInfo.size > 1 then
8          CGEstimator = DiffTriangle*(RefInfo)          /* Eq. (3.14) */
9      else if RefInfo.size == 1 then
10         CGEstimator = DiffTriangle(RefInfo)           /* Eq. (3.7) */
11     else
12         CGEstimator = CrMcs                    /* Eq. (3.5) */
13     LocEstimator = wMultilateration           /* subsection 3.5.2 */
       /* Phase I: Anchor Classification                                  */
14     CGPattern = CGRecognizer.filter(DGPattern)
15     CRPattern = CRRecognizer.filter(CGPattern)
       /* Phase II: Distance Estimation                                   */
16     DEs = ∅                                  /* the set of distance estimates */
17
18     foreach anchor j ∈ CRPattern do
19         DEs += [loc_j, CREstimator.estimate(anchor j), 0.2r]
20     foreach anchor j ∈ CGPattern do
21         DEs += [loc_j, CGEstimator.estimate(anchor j), 0.4r]
       /* Phase III: Location Estimation                                  */
22     return loc_i = LocEstimator.estimate(DEs)
```

Chapter 4

Range-Based Localization of Wireless Sensor Networks: An Accurate and Robust Approach

4.1 Overview

For wireless sensor networks (WSNs), it is important to derive the locations of sensor nodes from the distance measurements between neighboring nodes. The ranging techniques to measure distances can be divided into two categories according to accuracy. *Coarse-grained ranging techniques* have low accuracy at meter level because they exploit radio attenuation to measure distances, and these techniques include RSSI-based methods [Dimitrios et al., 2006, He et al., 2003] and proximity-based methods [Li and Liu, 2007, Lim and C., 2005, Niculescu and Nath, 2003, Wang and Xiao, 2006, Xiao et al., 2010a,b]. *Fine-grained ranging techniques* have high accuracy

at centimeter level since they adopt Time-Of-Arrival (TOA) technique, e.g. ultrasonic TOA or ultra-wideband TOA [Goldenberg et al., 2006, Horn et al., 1988, Ji and Zha, 2004, Meertens and Fitzpatrick, 2004, Moore et al., 2004, Priyantha et al., 2003, Savvides et al., 2003b, Wang et al., 2008]. This chapter adopts the fine-grained ranging techniques for accurate measurement of inter-node distances.

Existing solutions for the fine-grained localization can be divided into two categories: whole-topology approach and iterative approach. The *whole-topology approach* [Ji and Zha, 2004, Priyantha et al., 2003] directly analyzes the whole network topology by numerical optimization algorithms. For example, AFL [Priyantha et al., 2003] models the topology as a bunch of nodes connected by springs. AFL then relaxes these springs and let the location estimates of the nodes converge to their true locations guided by the spring forces. Another example is MDS-Map [Ji and Zha, 2004] which analyzes the network topology by multi-dimensional scaling. The major drawback of the whole-topology approach is that a network topology contains too many variables with each node having two variables x, y in a 2D space. The numerical tools that analyze a large topology directly can be easily trapped in local minima, especially when the topology is concave-shaped.

In contrast, the *iterative approach* [Goldenberg et al., 2006, Horn et al., 1988, Meertens and Fitzpatrick, 2004, Moore et al., 2004, Savvides et al., 2003b, Wang et al., 2008] divides the large network topology into small network elements, including individual nodes and groups of nodes (or patches). Each of the network elements has its own coordinate frame. Then among these network elements, the iterative approach picks out two elements, merges them to share a coordinate frame and generates a larger network element. Such a merging operation, since it manipulates only a small

number of variables (i.e. the relative position and orientation of the two network elements), can avoid the local minima problem which the whole-topology approach suffers from. For example, iterative multilateration [Savvides et al., 2003b] can align an individual node with a patch if the node has three distance measurements to nodes contained in the patch. Patch stitching [Horn et al., 1988] can align two patches if they share three nodes. CALL [Wang et al., 2008] can merge two patches if they share two nodes and are connected by a link, or if they share one node and are connected by two links, or if they are connected by four links. SWEEPS [Goldenberg et al., 2006] describes a scheme in which a sensor can be merged with a patch when the sensor is connected to the patch by only two links, but this sensor has two ambiguous positions relative to the patch. The ambiguity can be eliminated later when sufficient constraints are available.

However, these previous studies [Goldenberg et al., 2006, Horn et al., 1988, Moore et al., 2004, Savvides et al., 2003b, Wang et al., 2008] in the field of iterative localization are incomplete, in that they provide only parts of the sufficient conditions where two network elements can be merged (either uniquely or ambiguously). It is desirable to unify these sufficient conditions and give out a more robust condition that also considers previously neglected collinear geometry of sensor nodes. Besides the incomplete specification of conditions, there exist two other fundamental aspects that are not fully addressed. (1) When merging two network elements, how can ranging noise be tolerated, particularly for the merging of two patches? (2) How can flip ambiguities be enumerated? These flip ambiguities can be caused by roughly collinear nodes as interfered by ranging noise. The failure to enumerate these ambiguities may incur abnormally large localization error.

This chapter presents a problem of merging two network elements to form a larger one, in which a network element can either be an individual node or a group of nodes. This problem is named *body merging* problem (because we call a network element a body for short), which addresses two issues.

- Accuracy: align two bodies accurately by finding the best relative position and orientation of the two bodies that can minimize the mean squared error of their constraints. Note that such an optimal alignment can find only one of the possibilities of aligning two bodies.

- Robustness: enumerate all the possibilities of aligning two bodies, by discovering collinear geometry of nodes and enumerating flip ambiguities accordingly.

Providing a solution requires us to reveal the mutual dependency between the two issues of accuracy and robustness. (1) Robustness needs accuracy, because when accurately aligning two bodies we can obtain one of the ambiguities. By flipping this ambiguity across the line passing through collinear nodes, we can obtain all the other ambiguities. (2) Accuracy requires robustness, because without the proper handling of the robustness issue, the localization accuracy can be poor due to the unexpected flip ambiguities. Because this body merging can have finite ambiguities, we call a body as an *inflexible body* which has finite ambiguities.

Our solution to the inflexible body merging problem makes the following contributions.

- Accuracy: we propose an algorithm to merge two bodies optimally (i.e. to minimize the mean-squared error of their constraints and thus tolerate ranging noise). Our algorithm models the two bodies to be connected by springs and

then relaxes these springs to their minimum energy states. Directed by the spring forces, the bodies moves and their motions are modeled by the physical model of rigid body dynamics. Our algorithm is more generalized than traditional noise toleration algorithms, i.e. multilateration [Foy, 1976] and patch stitching [Horn et al., 1988]. This is because multilateration only aligns a node with a patch, and patch stitching can align two patches only when they shares three nodes. Our algorithm can align two bodies no matter how many nodes they share.

- Robustness: we present an algorithm to enumerate flip ambiguities during the merging of two bodies. The flip ambiguities are enumerated by flipping one of the ambiguities across the line passing collinear nodes during bodies merging. The challenge is that the previous work [Goldenberg et al., 2006] only considers the collinearity of anchor nodes during multilateration. But for body merging, sensor nodes can be collinear with anchor nodes, which produces flip ambiguities that are expected by previous work. Moreover, a set of collinear nodes can be just roughly collinear as interfered by ranging noise, and such an implicit collinearity may incur abnormally large localization error. But such an implicit collinearity can be detected by our algorithm based on orthogonal regression.

- Condition: we also provide a condition for the merging of two inflexible bodies. The basic idea is that the two bodies should have enough constraints by sharing nodes or having connecting links to confine their continuous relative motions (i.e. Degree-Of-Constraint $\geq$ Degree-Of-Freedom). This condition can unify previous work [Goldenberg et al., 2006, Savvides et al., 2003b, Wang et al., 2008] and help to achieve a higher localization percentage than state-of-the-art

CALL [Wang et al., 2008].

These declared contributions are validated by high-fidelity simulations with practical system parameters from [Moore et al., 2004, Savvides et al., 2003b].

The rest of this chapter is organized as follows. Section 5.4.2 introduces the background knowledge on network localization problem and iterative localization. Section 4.3 presents the body merging optimization problem and proposes an algorithm to align two bodies accurately by relaxing springs and based on rigid body dynamics. Section 4.4 presents the condition when two bodies can be merged uniquely and this condition also considers the collinear geometry of constraints between two bodies. Section 4.5 proposes the algorithm to enumerate flip ambiguities and ensure merging robustness. Section 4.6 shows our simulation results. Finally, Section 4.7 concludes this chapter.

4.2 Background Knowledge on Network Localization Problem and Iterative Localization

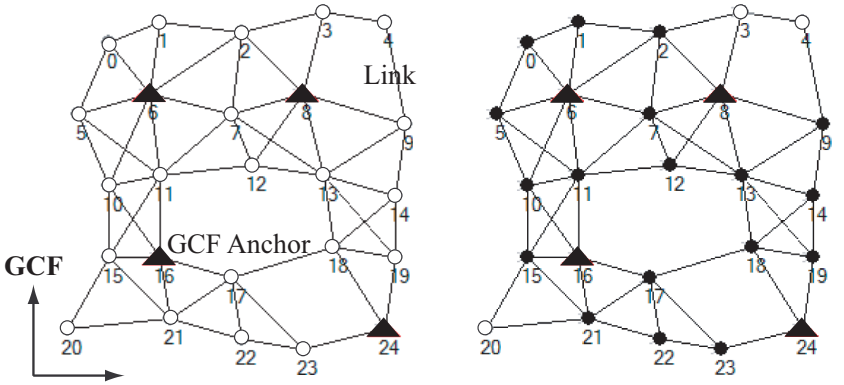
This section introduces the network localization problem, and describes the iterative localization approach which divides the network topology into bodies and merges them incrementally. This approach can simplify the network localization problem as a small-scale problem of merging two bodies.

Definition 1 (Network Localization Problem)**.**

Input*: distance measurements between neighboring nodes, and a small proportion of network nodes (called GCF anchors) whose locations are already known in the*

GCF (Global Coordinate Frame). The purpose of GCF anchors is to ensure the location estimates of normal nodes are defined in the GCF.

***Output**: location estimates of the nodes that can be localized uniquely. These location estimates should be defined in the GCF, whose purpose is to make these location estimates understandable for network users. For example, we can use the GPS coordinate frame as the GCF.*



(a) Input: Links and GCF Anchors (b) Output: Location Estimates

Fig. 4.1: Input and Output of Network Localization Problem.

***Example**: The input is illustrated in Fig. 4.1(a) where the GCF anchors are drawn as triangles and normal nodes are drawn as white dots. The output is shown in Fig. 4.1(b) where the location estimates are drawn as black dots. The nodes that do not have black dots in Fig. 4.1(b) have ambiguous location assignments. For example, node* 20 *does not have a black dot because it can flip across the line through nodes* $15, 21$.

Iterative Localization. Iterative localization approach solves network localization problem by the following two phases.

Phase 1 (Network Division). The output of network localization problem is location estimates in the same coordinate frame (i.e. the GCF). The iterative localization

approach therefore divides the network into a list of bodies, each of which has its own coordinate frame. As shown in Fig. 4.2, there are two types of bodies: *Global Body* whose coordinate frame is the GCF, and *Local Body* whose coordinate frame is a Local Coordinate Frame (LCF). There is only one global body, which contains all the GCF anchors with known locations in the GCF. There are numerous local bodies, each of which can be either a triangle or an individual node. For example, in Fig. 4.2, the three nodes k_1, k_2, k_3 form a triangle with edges $d_{k_1k_2}$, $d_{k_2k_3}$, $d_{k_3k_1}$. The LCF of this triangle can place its original point at node k_1, have x-axis through node k_2, have node k_3 above its x-axis. In Fig. 4.2, the node k_4 constitutes a local body, whose LCF can place its original point at k_4.

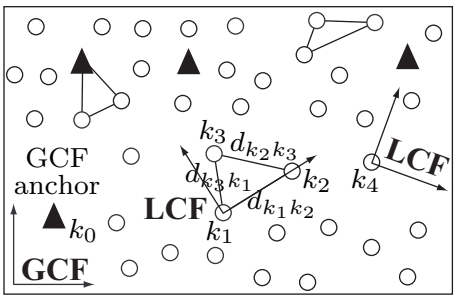
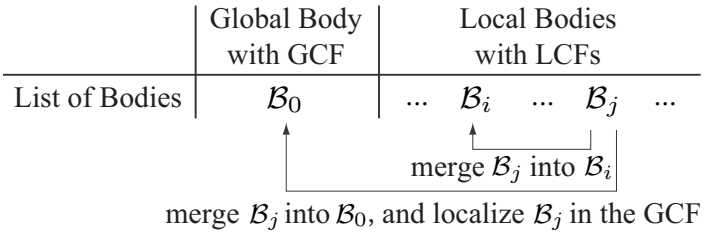


Fig. 4.2: Divide the Network Topology into Elementary Bodies.

Phase 2 (Iterative Body Merging). The network has been divided by phase 1 into a list of bodies [..., $\mathcal{B}_i$, ...], where

- $\mathcal{B}_i$ denotes the global body, if the index i is zero;

- $\mathcal{B}_i$ denotes a local body, if the index i is positive.



Phase 2 conquers the network localization problem by the proposed procedure in Algorithm 2 named Iterative Inflexible Body Merging (IIBM). This algorithm recursively selects two bodies from the body list (e.g. bodies $\mathcal{B}_i$ and $\mathcal{B}_j$ at Line 2) and merges them to share a coordinate frame (by Line 5). When this procedure cannot

Algorithm 2: IterativeInflexibleBodyMerging (IIBM)

Input: A List of Bodies $[\ldots, \mathcal{B}_i, \ldots]$ with index $i \geq 0$
Output: Global Body $\mathcal{B}_0$ with the GCF

```
1 begin
2     find two bodies in the body list that can be merged, namely body B_i and body
      B_j with 0 ≤ i < j
3     if no such B_i and B_j then return the global body B_0
4     else
5         merge body B_j into body B_i to generate a larger body B_i, and remove body
          B_j from the body list
6         goto Line 2
```

find two bodies that can be merged, it terminates at Line 3 and return the global body $\mathcal{B}_0$ whose nodes are localized in the GCF.

In Algorithm 2, Line 2 needs to tell whether two bodies can be merged, which is to be described in Section 4.4 for unique body merging and in Section 4.5.1 for body merging with finite ambiguities. Line 5 needs to solve how to merge two bodies, which is to be described in Section 4.3 for the accuracy issue (by tolerating ranging noise) and in Section 4.5.2 for the robustness issue (by enumerating flip ambiguities).

4.3 Accurate Body Merging against Noise

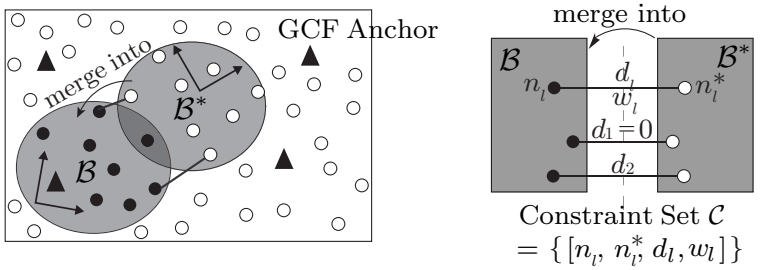
A challenge of aligning two bodies is the inevitable presence of ranging noise, which can degrade the body aligning accuracy. To tolerate the ranging noise, we propose the body merging optimization problem that minimizes the mean-squared error of the constraints between the two bodies. We then propose a solution to this problem, which is more generalized than the traditional noise toleration algorithms, i.e. multilateration [Foy, 1976] and patch stitching [Horn et al., 1988].

4.3.1 Body Merging Optimization Problem

Merging is the basic operation to merge two bodies, which requires a coordinate transformation between their coordinate frames. The challenge towards an accurate location transformation is the presence of ranging noises within the constraints that confine the relative motions of the two bodies. This subsection presents our algorithm to tolerate ranging noise and achieve an optimal transformation.

This chapter appears to be the first to summarize and solve the optimization problem (see Definition 3) to merge body $\mathcal{B}^*$ into body $\mathcal{B}$ as shown in Fig. 4.3(a). The traditional multilateration [Foy, 1976], which aligns a node with a patch, is just a special case of our algorithm where body $\mathcal{B}^*$ is a node and body $\mathcal{B}$ is a patch. Although patch stitching [Horn et al., 1988] can align two patches, it requires the two patches $\mathcal{B}$ and $\mathcal{B}^*$ to share three nodes. Our optimal body merging algorithm can align two patches when they share two nodes or less. Moreover, our algorithm can align two bodies for all situations in a unified manner.

It is assumed that the two bodies to merge have enough *constraints* by sharing nodes and having connecting links. As shown in Fig. 4.3(b), a constraint is drawn



(a) Merge Body $\mathcal{B}^*$ into Body $\mathcal{B}$ (b) Constraint Set $\mathcal{C}$

Fig. 4.3: Merging of Two Bodies $\mathcal{B}, \mathcal{B}^*$ with Constraint Set $\mathcal{C}$.

as an edge with one end incident to a node in body $\mathcal{B}$ (i.e. the black node) and the other end incident to a node in body $\mathcal{B}^*$ (i.e. the white node). Note that the length of the constraint can be zero when the constraint is a shared node (e.g. d_1 equals zero in Fig. 4.3(b)). These constraints confine the relative motions of the two bodies which thus can be aligned. A formal definition for the set of constraints is given in Definition 2.

Definition 2 (Constraint Set during Body Merging)**.** *During the merging of body $\mathcal{B}^*$ into body $\mathcal{B}$, the two bodies have the constraint set $\mathcal{C} = \{\ell_l\} = \{[n_l, n_l^*, d_l, w_l]\}$ as illustrated in Fig. 4.3(b), where*

- *ℓ_l is the l th ($0 \leq l < |\mathcal{C}|$) constraint contained in $\mathcal{C}$,*
- *n_l is node index of ℓ_l that is contained in body $\mathcal{B}$,*
- *n_l^* is node index of ℓ_l that is contained in body $\mathcal{B}^*$,*
  (Note: in all figures throughout the chapter, we draw node n_l as a black node and draw node n_l^* as a white node)
- *d_l is the measured length of the constraint ℓ_l, and*
- *w_l is the weight of the constraint ℓ_l.*

When a constraint ℓ_l corresponds to a node shared by body $\mathcal{B}$ and body $\mathcal{B}^*$, its length d_l is zero and its two node indices n_l, n_l^* are equal. Otherwise, a constraint ℓ_l is a link connecting the two bodies. Its two node indices n_l, n_l^* are different, and note that n_l, n_l^* should not be the indices of shared nodes.

The challenge for accurate body merging is the inevitable presence of ranging noise in these constraints' length estimation. We thus formalize a problem (called *body merging optimization problem*) in Definition 3 to tolerate ranging noise by minimizing the mean squared error in the constraints.

Definition 3 (Body Merging Optimization Problem)**.**
***Input**: The constraint set $\mathcal{C}' = \{\ell_l'\} = \{\,[p_l,\, p_l^*,\, d_l,\, w_l]\,\}$ confines the relative motion between body $\mathcal{B}$ and body $\mathcal{B}^*$. ℓ_l' is one of the constraints in set $\mathcal{C}'$, and ℓ_l' has four fields:*

- *p_l is the position of node n_l contained in body $\mathcal{B}$ and p_l is defined in body $\mathcal{B}$'s coordinate frame (see Fig. 4.4);*
- *p_l^* is the position of node n_l^* contained in body $\mathcal{B}^*$ and p_l^* is defined in body $\mathcal{B}^*$'s coordinate frame (see Fig. 4.4);*
- *d_l is the measured length of constraint ℓ_l';*
- *w_l is the weight to reflect d_l's measurement accuracy.*

***Manipulated Variables**: $T_{p,\,R}$ is a transformation function from body $\mathcal{B}^*$'s coordinate frame to $\mathcal{B}$'s coordinate frame:*

$$T_{p,\,R}\,(p^*) \quad = \quad p + R\,p^* \quad , \textit{ where} \tag{4.1}$$

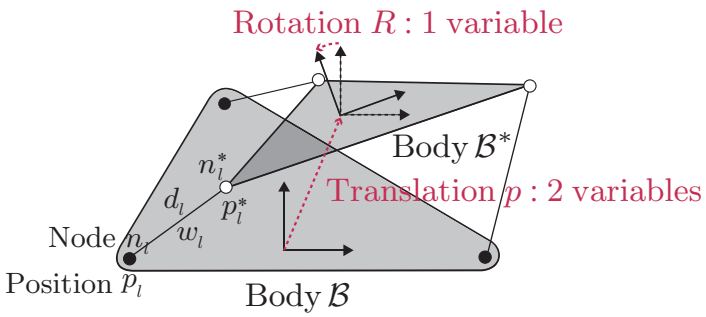


Fig. 4.4: Transformation $T_{p,R}$ with Three Variables in 2D.

- *p is position of body $\mathcal{B}^*$'s the original point in body $\mathcal{B}$'s coordinate frame as shown in Fig. 4.4, and*

- *R is rotation of body $\mathcal{B}^*$ about body $\mathcal{B}^*$'s original point.*

By function $T_{p,R}(p^*)$, a position p^* in body $\mathcal{B}^*$'s coordinate frame can be transformed to body $\mathcal{B}$'s coordinate frame.

***Objective**: The weighted error of constraint ℓ'_l is calculated as*

$$err_l(T_{p,R}) \quad = \quad w_l \left(d_l - \|p_l - T_{p,R}(p_l^*)\|\right) \quad , \text{ where}$$

- *$\|p_l - T_{p,R}(p_l^*)\|$ is the length of constraint ℓ'_l estimated from the two positions p_l and p_l^*, if given $T_{p,R}$.*

The objective function is the mean-squared error for all constraints in set $\mathcal{C}'$, which is noted as $f_{\mathcal{C}'}(T_{p,R})$.

$$f_{\mathcal{C}'}(T_{p,R}) \quad = \quad \sqrt{\frac{\sum_{\ell'_l \in \mathcal{C}'} err_l(T_{p,R})^2}{|\mathcal{C}'|}} \quad , \text{ where} \tag{4.2}$$

- *$|\mathcal{C}'|$ is the number of constraints in constraint set $\mathcal{C}'$.*

We use the symbol $T^{min}_{p,R}$ *to denote the optimal transformation function that can minimize mean-squared error* $f_{\mathcal{C}'}(T_{p,R})$.

$$T^{min}_{p,R} \quad = \quad \arg\min f_{\mathcal{C}'}(T_{p,R})$$

Restrictions*: For the optimized transformation* $T^{min}_{p,R}$*, the error of each constraint* ℓ'_l *has its magnitude below threshold* $2\,c\,\sigma$.

$$\forall\, \ell'_l \in \mathcal{C}' \quad err_l(T^{min}_{p,R}) < 2\,c\,\sigma \quad , \textit{ where} \tag{4.3}$$

- σ *is the expected ranging noise, and*
- c *is a constant which can be* 3 *for Gaussian noise.*

4.3.2 Optimal Body Merging Algorithm

For the body merging optimization problem in Definition 3, we propose a solution called *optimal body merging* algorithm. This algorithm recursively optimizes transformation function $T_{p,R}$ (or informally body $\mathcal{B}^*$'s spatial pose in the coordinate frame of body $\mathcal{B}$). The basic idea is to model each constraint as a spring as shown in Fig. 4.5. These springs due to their deformations cast their forces on body $\mathcal{B}^*$. This body $\mathcal{B}^*$ directed by these spring forces gradually moves towards the equilibrium and finally minimize the deformations of all springs. We simulate the movement of the body $\mathcal{B}^*$ using the physical model of *rigid body dynamics* [Witkin et al., 1997].

Although spring based modeling is also used by AFL [Priyantha et al., 2003], our solution has the following novelties. (1) AFL only simulates the translations of

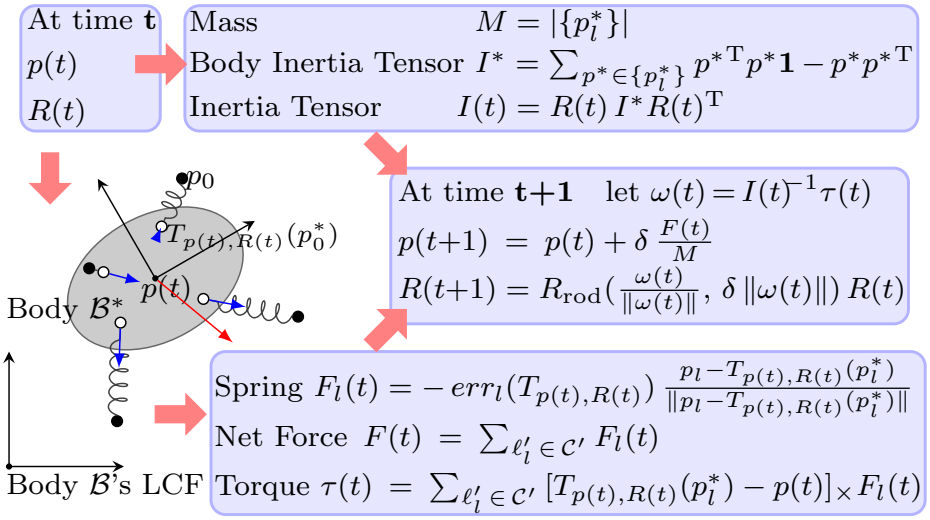


Fig. 4.5: Iterative Optimization based on Rigid Body Dynamics.

particles without volumes. In contrast, we simulate both translations and rotations of rigid bodies. Our solution thus can support both sensor merging where body $\mathcal{B}^*$ is an individual node and patch merging where body $\mathcal{B}^*$ contains more than one node. (2) AFL is suspectable to trap in local minima because AFL tries to solve a large-scale problem of optimizing a network of nodes. The Degree-Of-Freedom of such a large problem is the number of nodes multiplied by two in a 2D space, since each node has two translations separatively in x and y directions. In contrast, our solution can avoid such suboptimality problem because it only focuses on a small-scale problem to align two group of nodes, whose Degree-Of-Freedom is only three in a 2D space (with two translations and one rotation for the body $\mathcal{B}^*$).

Pseudocode. We present in Algorithm 3 the pseudocode of our optimal body merging algorithm. The output of Algorithm 3 is the optimal transformation function $T^{\min}_{p,R}$. With this output $T^{\min}_{p,R}$, the node positions in body $\mathcal{B}^*$ can be converted to the coordinate frame of body $\mathcal{B}$, and body $\mathcal{B}^*$ can thus be merged into body $\mathcal{B}$. The

inputs of Algorithm 3 are the constraint set $\mathcal{C}'$ (see Fig. 4.4) and the initial guesses of translation p and rotation R of the transformation $T^{\min}_{p,R}$.

Firstly, Line 2 asserts that the original point of $\mathcal{B}^*$'s coordinate frame has been moved to the centroid of all the white points in Fig. 4.4, or more precisely the set of points $\{p^*_l\}$ that are contained by body $\mathcal{B}^*$ and are incident to constraint set $\mathcal{C}'$. Such decentralization of body $\mathcal{B}^*$ is to make translation p of body $\mathcal{B}^*$ to be independent with its rotation R. This is because *for a rigid body, the translation of its centroid is independent from the rotation of this body about its centroid*, according to Euler's laws of motion [Witkin et al., 1997]. For the ease of extension to 3D spaces, we represent translation p by 3×1 vectors and represent rotation R by a 3×3 orthogonal matrix.

Algorithm 3: GetOptimizedTransfm (3D compatible)

Input: $\mathcal{C}' = \{\ell'_l\} = \{\ [p_l,\ p^*_l,\ d_l,\ w_l]\ \}$; initial guess of p, R
Output: transformation $T^{\min}_{p,R}$ with mean squared-error e

1 **begin**
2 **assert** centroid of $\mathcal{C}'$'s position set $\{p^*_l\}$ equals [0, 0, 0]
3 **repeat** `// start from guess` p, R `at time` $2t$
4 optimize guess p, R to guess $\_p$, $\_R$ at time $2t$+1
5 optimize guess $\_p$, $\_R$ to guess p, R at time $2t$+2
6 **until** *both* $\|p - \_p\|$ *and* $\|R - \_R\|_c$ *are small, where* $\|p - \_p\|$ *is magnitude of vector* $p - \_p$*, and* $\|R - \_R\|_c$ *is the maximum magnitude of all columns of matrix* $R - \_R$
7 **return** $T^{\min}_{p,R}$ with mean squared-error $e = f_{\mathcal{C}'}\,(T^{\min}_{p,R})$

At Line 4-6, the translation and rotation of body $\mathcal{B}^*$ are optimized iteratively. At the end of an iteration, current translation p and rotation R at time $2t$ are optimized to the new p, R at time $2t+2$, since p, R at time $2t$ is optimized by Line 4 to $\_p$, $\_R$ at time $2t$+1, and $\_p$, $\_R$ at time $2t$+1 is optimized by Line 5 to p, R at time $2t$+2. The loop terminates at Line 6 if the changes from time $2t+1$ to time $2t+2$ are negligibly small. Both Line 4 and Line 5 need to optimize translation $p(t)$ and

rotation $R(t)$ at time t to translation $p(t+1)$ and rotation $R(t+1)$ at time $t+1$, which is depicted in Fig. 4.5.

In Fig. 4.5, translation $p(t)$ at time t is optimized to $p(t+1)$ at time $t+1$. This optimization is directed by linear acceleration $F(t) \,/\, M$, where M is mass and $F(t)$ is net force combining all forces acting on body $\mathcal{B}^*$. When calculating the l^{th} spring force $F_l(t)$, weight w_l within $err_l(T_{p(t),R(t)})$ is modeled as the spring constant to capture the different error characteristics in different constraints. The parameter δ when calculating $p(t+1)$ is the optimization step size, whose configuration as a rule of thumb is $\delta \in [0.5a,\, a]$ $(a = M \,/\, \sum_{\ell'_l \in C'} w_l)$ to balance between convergence speed and potential oscillations.

In Fig. 4.5, rotation $R(t)$ at time t is optimized to $R(t+1)$. This optimization is directed by angular acceleration $\omega(t) = I(t)^{-1}\,\tau(t)$, where $I(t)$ is inertial tensor and $\tau(t)$ is torque. With this $\omega(t)$, $R(t+1)$ can be calculated by Rodrigues' rotation formula $R_{\text{rod}}(\phi,\, \theta)$ that rotates $R(t)$ around unit vector ϕ $(\phi = \omega(t) \,/\, \|\omega(t)\|)$ by angle θ $(\theta = \delta\, \|\omega(t)\|)$. After this

$$R_{\text{rod}}(\phi,\, \theta) \;=\; I \,+\, \sin\theta\, [\phi]_\times \,+\, (1 \,-\, \cos\theta)\, [\phi]_\times^2$$

rotation, orthogonality of $R(t+1)$ should be maintained, since we represent rotations by 3×3 rotation matrices, which compared with quaternion has numerical drift problem [Witkin et al., 1997]. But rotation matrix is easier to manipulated by standard matrix operations. When calculating the torque $\tau(t)$ acting on body $\mathcal{B}^*$, operator $[\,.\,]_\times$ converts vector a to its skew-symmetric matrix $[a]_\times$, in order to transform vector

cross product to

$$\text{Given vector } a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix},\ [a]_\times \stackrel{\text{def}}{=\!=} \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}.$$

matrix multiplication (i.e. $a \times b = [a]_\times\, b$). Although the equations in Fig. 4.5 are defined in 3D, they can easily handle the cases in 2D by setting z value of each position to zero and by setting each orientation vertical to z-axis.

4.3.3 Two Technical Issues

Conquer Local Minima Problem. Our optimal body merging algorithm (i.e. Algorithm 3) is essentially a greedy optimization algorithm, which in practice has a non-negligible possibility to trap in local minima. The traditional Multilateration also has this kind of problem. We can conquer this local minima problem by the following strategy: generate a set of initial guesses of the optimal transformation function $T^{\min}_{p,R}$, then separatively optimize them by Algorithm 3, and finally select the best solution with the minimum mean-squared error (see Eq. (4.2) for the definition of mean-squared error).

We describe a method to generate a set of good-quality initial guesses about the optimal transformation function $T^{\min}_{p,R}$.

(1) We generate the guess of translation p in $T^{\min}_{p,R}$ by the extended bounding box algorithm as follows. A bounding box can be obtained in Fig. 4.6 that corresponds to a constraint with length d_l and with two vertices p_l, p^*_l. The position p_l is de-

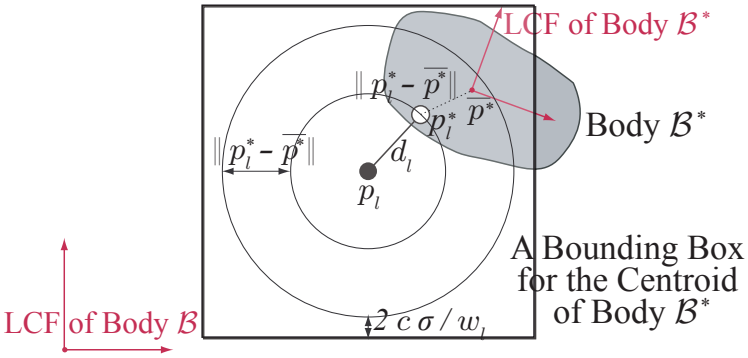


Fig. 4.6: A Bounding Box for Centroid $\overline{p^*}$ of Body $\mathcal{B}^*$.

fined in body $\mathcal{B}$'s coordinate frame and position p_l^* is defined in body $\mathcal{B}^*$'s coordinate frame. The centroid of body $\mathcal{B}^*$ is denoted as $\overline{p^*}$. Then this centroid $\overline{p^*}$ should be included in the bounding box centered at position p_l with its radius to be $d_l + \|p_l^* - \overline{p^*}\| + 2\,c\,\sigma\,/\,w_l$, where $\|p_l^* - \overline{p^*}\|$ is distance between position p_l^* and the centroid $\overline{p^*}$ and $2\,c\,\sigma\,/\,w_l$ is the upper bound of measurement error of constraint length d_l. Each constraint in set $\mathcal{C}'$ that connects the two bodies can provide such a bounding box for the centroid of body $\mathcal{B}^*$. Then the guesses of the position of this centroid can be generated within the overlapped box of all the bounding boxes.

(2) we generate a guess of rotation R randomly with Euler angle $\gamma \in (-\pi, \pi]$. Moreover, rotation matrix R should be randomly multiplied with a reflection matrix.

Mitigate Error Accumulation by Refinement. The body merging optimization problem in Definition 3 is focused on minimizing the mean-squared error within the constraints whose length estimates are noisy. However, this problem formulation neglects that the position assignments of a body can also be inaccurate with noise. More specifically, in Fig. 4.4, both the constraint length measurement d_l and the two positions p_l, p_l^* can be inaccurate. To address this subtle issue, an optional refinement step can be added after the bodies are merged by Algorithm 3. This refinement step

(as shown at Line 4 of Algorithm 4) decomposes the body merged from $\mathcal{B}$ and $\mathcal{B}^*$ (i.e. body $\mathcal{B}^+$ constructed at Line 3) into a network of nodes connected by springs, and apply AFL [Priyantha et al., 2003] to this network of nodes to further refine their positions (see Line 4). Finally, this refined body $\mathcal{B}^+$ is verified at Line 5 to check whether the error of each constraint is below a threshold (see Eq. (4.3)). If the verification fails, Algorithm 3 declares the failure to merge two bodies by returning an empty set.

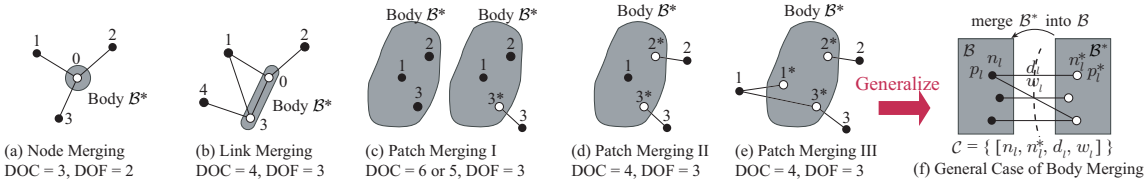
Algorithm 4: MergeTwoBodiesOptimally

Input: Bodies $\mathcal{B}$,$\mathcal{B}^*$; Constraint Set $\mathcal{C} = \{\ell_l\} = \{\,[n_l,\, n_l^*,\, d_l,\, w_l]\,\}$
Output: Body $\mathcal{B}^+$ by merging body $\mathcal{B}^*$ into body $\mathcal{B}$

1 **begin**
2 Get the optimal transformation function $T_{p,\,R}^{\min}$ from $\mathcal{B}^*$'s coordinate frame to $\mathcal{B}$'s coordinate frame (by Algorithm 3)
3 Get body $\mathcal{B}^+$ by merging the node coordinates in $\mathcal{B}$ with the node coordinates in $\mathcal{B}^*$ transformed by $T_{p,\,R}^{\min}$
4 (*optional*) Refine this body $\mathcal{B}^+$ by AFL [Priyantha et al., 2003]
5 Verify $\mathcal{B}^+$ by testing whether the error of each constraint is below a threshold (see Eq. (4.3)), and if it fails **return** $\emptyset$

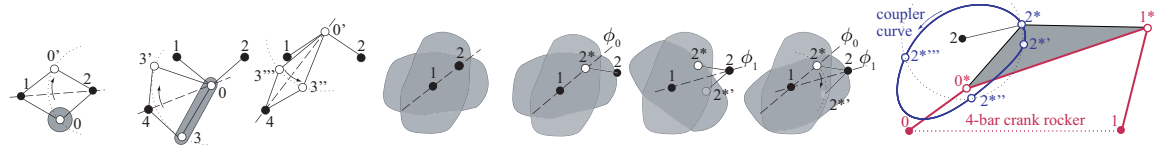
4.4 Unique Body Merging Conditions

This section presents two necessary conditions for the unique alignment of two bodies. The first condition is the redundancy in the constraints between two bodies. This condition can unify the previous work in the field of global rigidity [Horn et al., 1988, Savvides et al., 2003b, Wang et al., 2008]. The second condition is the non-collinear geometry of given constraints. This non-collinearity is not fully explored by the previous work [He et al., 2003], due to the possibility that the black nodes and white nodes can also be collinear during body merging, as illustrated in Fig. 4.9.



(a) Node Merging DOC = 3, DOF = 2 (b) Link Merging DOC = 4, DOF = 3 (c) Patch Merging I DOC = 6 or 5, DOF = 3 (d) Patch Merging II DOC = 4, DOF = 3 (e) Patch Merging III DOC = 4, DOF = 3 (f) General Case of Body Merging

Fig. 4.7: Unique body merging whose constraints are redundant with DOC > DOF. A constraint, as shown in (f), is a link that connects a black node in body $\mathcal{B}$ with a white node in body $\mathcal{B}^*$. When the length of a constraint is zero, the constraint is drawn as a black node on body $\mathcal{B}^*$, e.g. in (c). We have listed all the possible cases in (a-e): in (a), body $\mathcal{B}^*$ contains one node with label 0; in (b), body $\mathcal{B}^*$ is a link that contains two nodes $0, 3$; in (c-e), body $\mathcal{B}^*$ contains at least three nodes.



(a) Node Merging DOC = 2, DOF = 2 (b) Link Merging DOC = 3, DOF = 3 (c) Patch Merging I DOC = 4, DOF = 3 (d) Patch Merging II DOC = 3, DOF = 3 (e) Patch Merging III DOC = 3, DOF = 3

Fig. 4.8: Ambiguous body merging with finite ambiguities due to non-redundant constraints with DOC $\geq$ DOF. These constraints are insufficient for unique body merging. But they can restrict the continuous motions of body $\mathcal{B}^*$: in (a), node 0 cannot continuously rotate around node 2 due to the restriction of link [1,0]; in (b), node 3 cannot freely rotate around node 0; in (c)(d), body $\mathcal{B}^*$ cannot rotate freely around node 1; in (f), the node 2^* of body $\mathcal{B}^*$ cannot slide freely along the coupler curve.

4.4.1 Unique Body Merging Condition: Redundant Constraints

For the unique merging of two bodies, a necessary condition is the redundancy in their constraints, which is formulated as

$$\text{Degree-Of-Constraint (DOC)} \;>\; \text{Degree-Of-Freedom (DOF)} \,.$$

- *Calculation of DOF*: if body $\mathcal{B}^*$ contains only one node, DOF is two; otherwise, DOF is three due to the rotation of $\mathcal{B}^*$.

- *Calculation of DOC*: in constraint set $\mathcal{C}$, each shared node can contribute two

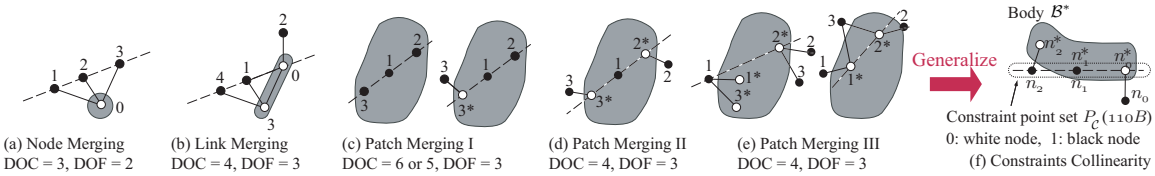


Fig. 4.9: Ambiguous body merging due to constraints collinearity, i.e. the existence of the depicted line that passes through the nodes of each constraint like in (f). Thus body $\mathcal{B}^*$ can flip across the depicted lines and preserve constraint length.

DOC, and each connecting link can contribute one DOC. The DOC value for each case in shown in Fig. 4.7.

This necessary condition can unify the previous work [Horn et al., 1988, Savvides et al., 2003b, Wang et al., 2008] that localizes individual nodes and patches. We explain this condition case by case as follows.

Case (a). Trilateration is an operation that merges body $\mathcal{B}^*$ containing a single node into body $\mathcal{B}$. As shown in Fig. 4.7(a), body $\mathcal{B}$ contains the three nodes $1, 2, 3$ which are drawn as black nodes. Body $\mathcal{B}^*$ contains only the node 0 which is drawn as a white node. Body $\mathcal{B}^*$ has two DOF in the coordinate frame of body $\mathcal{B}$, because node 0 can move in two directions, i.e. x and y directions. Trilateration requires at least three links connecting body $\mathcal{B}^*$ to body $\mathcal{B}$, because the three links can provide three DOC to redundantly constrain the two DOF of body $\mathcal{B}^*$. Otherwise, as shown in Fig. 4.8(a), node 0 can have an ambiguous position assignment $0'$, if there are just two links $[0, 1]$, $[0, 2]$ providing two DOC.

Case (b). Collaborative Multilateration [Savvides et al., 2003b] is an operation that merges body $\mathcal{B}^*$ containing two nodes into body $\mathcal{B}$. As shown in Fig. 4.7(b), body $\mathcal{B}$ contains the three black nodes $1, 2, 4$. Body $\mathcal{B}^*$ contains the two white nodes $0, 3$. Because body $\mathcal{B}^*$ contains more than one node, it has three DOF with two translations

and one rotation. Collaborative Multilateration requires four links connecting body $\mathcal{B}^*$ to body $\mathcal{B}$, which can provide four DOC and redundantly constrain the three DOF. Otherwise, as shown in Fig. 4.8(b), the body $\mathcal{B}^*$ can have four spatial poses, i.e. $[0,3]$, $[0,3']$, $[0',3'']$ and $[0',3''']$, if there are just three links providing three DOC.

Case (c). Patch Stitching [Horn et al., 1988] is an operation that merges body $\mathcal{B}^*$ containing at least three nodes into body $\mathcal{B}$. As shown in Fig. 4.7(c), body $\mathcal{B}$ contains the three black nodes $1,2,3$. Body $\mathcal{B}^*$ is drawn as a gray block. Patch stitching requires body $\mathcal{B}$ and $\mathcal{B}^*$ share at least three nodes which are drawn as three black nodes $1,2,3$ fixing on body $\mathcal{B}^*$. Different from a link that can provide one DOC, each shared node can provide two DOC. This is because body $\mathcal{B}^*$ by sharing a node with body $\mathcal{B}$ can only rotate about the shared node, which indicates the two translations body $\mathcal{B}^*$ is constrained by the shared node.

Case (d)(e). CALL [Wang et al., 2008] proposed two other conditions that two patches can be merged uniquely, as shown in Fig. 4.7(d)(e). Fig. 4.7(d) depicts the condition of one shared node and two connecting links, which can provide four DOC. Fig. 4.7(e) shows the condition of four connecting links, which can provide four DOC. If DOC is not redundant and is equal to DOF, the body $\mathcal{B}^*$ can have multiple spatial poses. In Fig. 4.8(d), the body $\mathcal{B}^*$ has four ambiguities, if we flip body $\mathcal{B}^*$ across the depicted lines. In Fig. 4.8(e), the body $\mathcal{B}^*$ can have four ambiguities when body $\mathcal{B}^*$ is connected to $\mathcal{B}$ by only three links $[0,0^*]$, $[1,1^*]$ and $[2,2^*]$. The upper bound for the number of ambiguities for Fig. 4.8(e) is twelve, which is to be proved in Section 4.5.1.

4.4.2 Unique Body Merging Condition: Geometry of Constraints

Motivation. It is well-known that for multilateration the three black nodes $1, 2, 3$ in Fig. 4.7(a) should be non-collinear. Otherwise, as shown in Fig. 4.9(a), the node 0 can flip across the depicted line without changing the length of each constraint. However, the non-collinearity during body merging is still not thoroughly explored by previous work. This is because multilateration only considers the collinearity of black nodes, and it is possible that black nodes and white nodes are collinear during body merging. For example, in Fig. 4.9(b), the nodes $0, 1, 4$ are collinear and node 3 can flip across the depicted line; in Fig. 4.9(c), the body $\mathcal{B}^*$ can flip across the line through the nodes $1, 2, 3^*$.

We propose a necessary condition for unique body merging, which can fully explore the collinear geometry of the constraint set $\mathcal{C}$ (see Definition 2 about constraint set). The basic idea is that there does not exist a line that can pass through a vertex of each constraint. Otherwise, body $\mathcal{B}^*$ can flip across this line and has ambiguous realizations as shown in Fig. 4.9.

To describe the point set that contains a vertex of each constraint in constraint set $\mathcal{C}$, we define the concept of *constraint point set* in Definition 4 and with the notation of $P_{\mathcal{C}}(k)$, where k is an integer. For example, in Fig. 4.9(f), constraint set $\mathcal{C}$ contains three constraints: one of them is a shared node n_1 (equal to n_1^*) and the other two are links $[n_0, n_0^*]$, $[n_2, n_2^*]$. The constraint point set $P_{\mathcal{C}}(110B)$ contains three nodes indices n_2, n_1, n_0^*, because k is equal to the binary $110B$. $P_{\mathcal{C}}(110B)$ can also be written as $P_{\mathcal{C}}(6)$ since binary $110B$ is equal to decimal 6.

Definition 4 (Constraint Point Set $P_{\mathcal{C}}(k)$)**.** *Assume that* $\mathcal{C} = \{\ell_l\} = \{\, [n_l,\, n_l^*,\, d_l,\, w_l] \,\}$

is the constraint set during the merging of body $\mathcal{B}^*$ into body $\mathcal{B}$, as shown in Fig. 4.7(f). The constraint point set $P_{\mathcal{C}}(k)$ corresponding to $\mathcal{C}$ contains one of the nodes of each constraint ℓ_l ($\in \mathcal{C}$). The integer k, which is between 0 and $2^{|\mathcal{C}|}$ (exclusive), can indicate which node (n_l or n_l^*) is contained by $P_{\mathcal{C}}(k)$. If the l th bit of the integer k is 1, point set $P_{\mathcal{C}}(k)$ contains the node n_l in constraint ℓ_l; otherwise point set $P_{\mathcal{C}}(k)$ contains the node n_l^* in constraint ℓ_l.

With the concept of constraint point set $P_{\mathcal{C}}(k)$, we present in Theorem 1 a necessary condition for the unique merging of two bodies: the non-collinear geometry of constraint set $\mathcal{C}$. This condition fully considers the collinearity of different combinations of black nodes and white nodes shown in Fig. 4.9, by varying integer k to generate different node combinations.

Theorem 1 (Constraint Set Non-collinearity)**.** *For the unique merging of body $\mathcal{B}^*$ into body $\mathcal{B}$, a necessary condition is the non-collinearity of constraint set $\mathcal{C}$ as defined below:*

- *when body $\mathcal{B}^*$ contains one node, constraint point set $P_{\mathcal{C}}(k)$ is non-collinear with $k = 2^{|\mathcal{C}|} - 1$ (or $11..1B$);*
- *when body $\mathcal{B}^*$ contains two nodes, constraint point set $P_{\mathcal{C}}(k)$ is non-collinear for each $k \in [1,\ 2^{|\mathcal{C}|} - 1]$;*
- *when body $\mathcal{B}^*$ contains three nodes or more, constraint point set $P_{\mathcal{C}}(k)$ is non-collinear for each $k \in [0,\ 2^{|\mathcal{C}|} - 1]$.*

Proof. When body $\mathcal{B}^*$ contains one node as depicted in Fig. 4.9(a), the three black nodes (denoted as $P_{\mathcal{C}}(2^{|\mathcal{C}|} - 1)$) should be non-collinear for the unique merging. When

body $\mathcal{B}^*$ contains two nodes as depicted in Fig. 4.9(b), the two white nodes (denoted as $P_{\mathcal{C}}(0)$) can be collinear because the flipping of body $\mathcal{B}^*$ across the line through $P_{\mathcal{C}}(0)$ cannot change the locations of the two nodes in $\mathcal{B}^*$. But other $P_{\mathcal{C}}(k)$ with $k \in [1,\ 2^{|\mathcal{C}|} - 1]$ should be non-collinear for the unique merging. When $\mathcal{B}^*$ contains at least three nodes as depicted in Fig. 4.9(c-e), $P_{\mathcal{C}}(k)$ with $k \in [0,\ 2^{|\mathcal{C}|} - 1]$ should be non-collinear. Otherwise, the flipping across the line passing through collinear $P_{\mathcal{C}}(k)$ can change node locations of $\mathcal{B}^*$. □

Collinearity Testing Problem. A key issue required by Theorem 1 is the ability to test whether a constraint point set $P_{\mathcal{C}}(k)$ is collinear. The challenge is the presence of ranging noise, because the constraint point set $P_{\mathcal{C}}(k)$ can be roughly collinear as interfered by ranging noise.

To test the rough collinearity of the given constraint point set $P_{\mathcal{C}}(k)$, our basic idea is to

1. obtain by orthogonal regression a line ϕ fitting the constraint point set $P_{\mathcal{C}}(k)$, as shown in Fig. 4.10, and

2. test whether the distance from each point to line ϕ is within a threshold that relates with ranging noise σ.

Firstly, we apply orthogonal regression algorithm (i.e. Algorithm 5) to the constraint point set $P_{\mathcal{C}}(k)$ and obtain a line ϕ that fits the coordinates of $P_{\mathcal{C}}(k)$. For example, in Fig. 4.10, the constraint point set $P_{\mathcal{C}}(k)$ contains seven node indices. Four of them are black nodes (i.e. n_0, n_3, n_4, n_5), which are contained in body $\mathcal{B}$. Three of them are white nodes (i.e. n_1^*, n_2^*, n_6^*), which are contained in body $\mathcal{B}^*$. We can apply orthogonal regression to the seven nodes to obtain the fitted line ϕ.

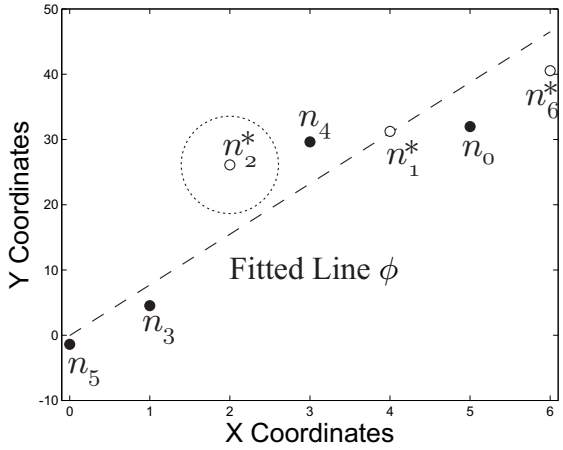


Fig. 4.10: Test Collinearity of Constraint Point Set $P_{\mathcal{C}}(k)$.

Although the positions of the seven nodes are defined in different coordinate frames, we can transform the coordinates of the white nodes to the coordinate frame of body $\mathcal{B}$, because we have the coordinate transformation function obtained by Algorithm 3 to transform body $\mathcal{B}^*$'s coordinate frame to body $\mathcal{B}$'s coordinate frame. Thus all the seven nodes can have their coordinates in the same coordinate frame.

Algorithm 5: Orthogonal Regression (or total least squares)

Input: a set of data points $\{ [x_l,\ y_l,\ w_l] \}$ in the same coordinate frame, where w_l is the weight of point x_l, y_l

Output: fitted line ϕ: $Ax + By + C = 0$ that minimizes the orthogonal distances from the data points to line ϕ

begin

$\bar{x} = \sum w_l^2 x_l \,/\, \sum w_l^2; \quad \bar{y} = \sum w_l^2 y_l \,/\, \sum w_l^2$

$x'_l = x_l - \bar{x};\ y'_l = y_l - \bar{y};\ W = \begin{bmatrix} \sum w_l^2 x'_l x'_l & \sum w_l^2 x'_l y'_l \\ \sum w_l^2 y'_l x'_l & \sum w_l^2 y'_l y'_l \end{bmatrix}$

$[A, B]^{\mathrm{T}}$ is equal to the eigenvector of the matrix W with the minimum eigenvalue, and $C = -A\,\bar{x} - B\,\bar{y}$

Secondly, we calculate the distance from each point in $P_{\mathcal{C}}(k)$ to ϕ. If each distance is below the threshold $2\,c\,\sigma/w_l$, the constraint point set $P_{\mathcal{C}}(k)$ is regarded to be collinear; otherwise, the constraint point set $P_{\mathcal{C}}(k)$ is non-collinear. Here, σ/w_l is the

measurement noise of constraint ℓ_l and c is an adjustable constant. For example, in Fig. 4.10, the seven points are non-collinear because the distance of the node n_2^* to the fitted line ϕ is larger than the threshold.

4.5 Body Merging with Finite Ambiguities

In a sparse network, the redundancy in constraints (i.e. DOC > DOF) for unique body merging may not always be satisfied. This section thus considers *non-underdetermined* constraints (i.e. DOC $\geq$ DOF), with which two bodies can be merged with finite ambiguities as shown in Fig. 4.8. This section firstly proves that non-underdetermined constraints are the sufficient condition for the body merging with finite ambiguities, and then presents our algorithm to enumerate flip ambiguities to improve the robustness of body merging.

When this idea of finite ambiguities is incorporated into the iterative body merging algorithm in Algorithm 2, we call it *Iterative Inflexible Body Merging* (IIBM) where an inflexible body without any flexible parts can have finite ambiguities. Compared with the state-of-the-art SWEEPS [Goldenberg et al., 2006] and CALL [Wang et al., 2008] that also adopts the idea of finite ambiguities, our IIBM algorithm has two advantages. (1) Better robustness of body merging against collinear constraints: This is because the collinear geometry of constraints can also produce finite ambiguities as shown Fig. 4.9. Such flip ambiguities need to be enumerated, which is to be described in Section 4.5.2. (2) Higher percentage of localizable nodes: This is because IIBM applies all the cases, including the unique merging cases in Fig. 4.7 and the ambiguous merging cases in Fig. 4.8, equally to

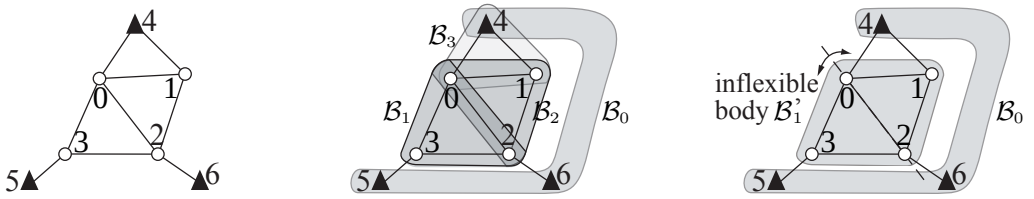
- the merging of a local body into the global body and

- the merging of a local body into another local body,

In contrast, CALL [Wang et al., 2008] does not use the ambiguous cases in Fig. 4.8(c)(d) for the merging of two local bodies.

An Example of IIBM algorithm. The network in Fig. 4.11(a) needs to be localized, where nodes $4, 5, 6$ are GCF anchors nodes in global coordinate frame. Firstly, the network is divided into a list of bodies shown in Fig. 4.11(b), including

- global body $\mathcal{B}_0$ containing the three GCF anchors $4, 5, 6$,
- local body $\mathcal{B}_1$ (i.e. the triangle with nodes $0, 2, 3$),
- local body $\mathcal{B}_2$ (i.e. the triangle with nodes $0, 1, 2$),
- local body $\mathcal{B}_3$ (i.e. the triangle with nodes $0, 1, 4$).



(a) Input Network (b) Network Division (c) Merge Body $\mathcal{B}_1'$ into $\mathcal{B}_0$

Fig. 4.11: An Example of Iterative Inflexible Body Merging.

Then the bodies are merged iteratively. The local body $\mathcal{B}_1$ and the local body $\mathcal{B}_2$ in Fig. 4.11(b) share two nodes 0 and 2, which corresponds to the ambiguous body merging case in Fig. 4.8(c). Thus they can be merged with two ambiguities, and generate the inflexible body $\mathcal{B}_1'$ in Fig. 4.11(c) which can be folded across the line through nodes $0, 2$. The inflexible body $\mathcal{B}_1'$ and the global body $\mathcal{B}_0$ have four connecting links, which corresponds to the case in Fig. 4.7(e). Thus inflexible body $\mathcal{B}_1'$

can be localized in GCF. The wrong ambiguity of inflexible body $\mathcal{B}_1'$ can be rejected, because when the wrong ambiguity of $\mathcal{B}_1'$ is aligned with the global body $\mathcal{B}_0$ (by Algorithm 3), the two bodies cannot fit with each other. That is no matter how we adjust the spatial pose of the wrong ambiguity of body $\mathcal{B}_1'$, the error of all constraints cannot be reduced below the defined threshold (see Line 5 of Algorithm 5).

4.5.1 Condition of Non-underdetermined Constraints

The theoretical foundation of inflexible body merging is that there are only finite ambiguities during the merging of body $\mathcal{B}^*$ into body $\mathcal{B}$ when the constraint set $\mathcal{C}$ is non-underdetermined with DOC $\geq$ DOF, which is illustrated in Fig. 4.7. For this theoretical foundation, we provide a formal proof based on quadratic equation systems and Bézout's theorem. Moreover, with this proof, a tight upper bound of 12 is derived for the number of ambiguities when the two bodies are connected by three links as shown in Fig. 4.8(e) (see Theorem 4). This upper bound of 12 is much tighter than the upper bound of 24 provided by CALL [Wang et al., 2008].

We firstly present the governing equations in Eq. (4.4) which have three unknowns in 2D. Two unknowns is contained in translation p and one unknown is in rotation R. The real solutions to this parallel equation set are all the ambiguities that two bodies can be aligned. Note that Eq. (4.4) can lose its rotation parameter R and degrades to multilateration, when body $\mathcal{B}^*$ contains only one node and thus all sensors overlap as $p_l^* \equiv \mathbf{0}$. Thus for the following analysis we assume the nodes p_l^* do not overlap with at least one of $\|p_{l_1}^* - p_{l_2}^*\|$ $(0 \leq l_1 < l_2 < n)$ to be nonzero.

Definition 5 (Governing Equations of Body Merging Problem)**.** *During the merging of body $\mathcal{B}^*$ into body $\mathcal{B}$, we have the constraint set $\mathcal{C}' = \{\ell_l'\} = \{\,[p_l,\, p_l^*,\, d_l,\, w_l]\,\}$ as*

defined in Definition 3. The purpose is to find all the real roots of the following governing equation system.

$$d_l^2 \quad = \quad \|(p + R\,p_l^*) - p_l\|^2 \quad (0 \le l < |\mathcal{C}'|) \tag{4.4}$$

For simplicity, Eq. (4.4) omits the reflection of body $\mathcal{B}^*$, which can later be implemented by left multiplying each position p_l^* of body $\mathcal{B}^*$ by matrix $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$.

We convert Eq. (4.4) to the equation system about translation p and rotation t in Eq. (4.5), where t is equal to $\tan\frac{\gamma}{2}$ and γ is the rotational angle of the rotation matrix R. Rotation R in Eq. (4.4) can be substituted by rotation matrix with angle γ.

$$\begin{aligned} d_l^2 &= \| \begin{bmatrix} \cos\gamma & \sin\gamma \\ -\sin\gamma & \cos\gamma \end{bmatrix} p_l^* + (p - p_l) \|^2 \\ &= \| \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} p_l^* \cos\gamma + \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} p_l^* \sin\gamma + (p - p_l) \|^2 \\ &= \|p_l^* \cos\gamma + \vec{p}_l^{\,*} \sin\gamma + (p - p_l)\|^2 \\ &= \|p_l^*\|^2 + \|p - p_l\|^2 + 2p_l^* \cdot (p - p_l)\cos\gamma + 2\vec{p}_l^{\,*} \cdot (p - p_l)\sin\gamma \end{aligned}$$

where $\vec{p}_l^{\,*} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} p_l^*$. Note that $p_l^* \cdot \vec{p}_l^{\,*} \equiv 0$, $\|p_l^*\| \equiv \|\vec{p}_l^{\,*}\|$. Simplification is achieved by introducing symbols A_l, B_l, C_l.

$$\begin{aligned} 0 &= A_l + B_l \cos\gamma + C_l \sin\gamma \\ A_l &= \|p\|^2 - 2\,p_l \cdot p + \|p_l\|^2 + \|p_l^*\|^2 - d_l^2 \\ B_l &= 2\,p_l^* \cdot p - 2\,p_l^* \cdot p_l \qquad C_l \;=\; 2\,\vec{p}_l^{\,*} \cdot p - 2\,\vec{p}_l^{\,*} \cdot p_l \end{aligned}$$

The above equations can be transformed to the polynomial system below about $t = \tan\frac{\gamma}{2}$ by tangent half-angle formulae.

$$\mathbf{0} \quad = \quad \begin{bmatrix} A_l + B_l & 2\,C_l & A_l - B_l \end{bmatrix} \begin{bmatrix} t^0 & t^1 & t^2 \end{bmatrix}^{\mathrm{T}} \tag{4.5}$$

By solving the equation system in Eq. (4.5), we give out in Theorem 2-4 the upper bounds of the number of ambiguities for different cases with DOC $\geq$ DOF.

Theorem 2. *There are at most four ambiguities during the merging of body $\mathcal{B}^*$ into body $\mathcal{B}$, if*

- *they can satisfy the condition of DOC $\geq$ DOF, and*
- *they share at least one node as shown in Fig. 4.8(d).*

Proof. The following two parallel equations can be established from the two constraints in Fig. 4.8(d).

$$\begin{aligned} 0 \quad &= \quad d_0^2 \quad = \quad \|p - p_0\|^2 \qquad (\text{with } d_0 = 0 \text{ and } p_0^* = \mathbf{0}) \\ 0 \quad &= \quad \begin{bmatrix} A_1 + B_1 & 2\,C_1 & A_1 - B_1 \end{bmatrix} \begin{bmatrix} t^0 & t^1 & t^2 \end{bmatrix}^{\mathrm{T}} \end{aligned}$$

The first equation corresponds to the zero-length link with $d_0 = 0$ and it is additionally assumed $p_0^* = 0$, which can be realized by a translation of body $\mathcal{B}^*$'s coordinate frame. From the first equation, it can be known that translation p is equal to p_0. From the second equation which follows Eq. (4.5), 2 roots can be derived for variable t, since p is already known and the second equation is quadratic in t. With a known t, we have only one γ and thus only one rotation R, since $t = \tan\frac{\gamma}{2}$. Considering the reflection

of body $\mathcal{B}^*$, which can double the number of ambiguities, variables p, R can have at most 4 solutions and thus Fig. 4.8(d) can have 4 ambiguities. □

Theorem 3. *There are at most eight ambiguities during the merging of body $\mathcal{B}^*$ into body $\mathcal{B}$, if*

- *they can satisfy the condition of DOC $\geq$ DOF,*
- *they does not share a node as shown in Fig. 4.8(e), and*
- *a node of body $\mathcal{B}^*$ is incident to two links, e.g. $p_0^* = p_1^*$.*

Proof. The following two parallel equations can be established from the three links in Fig. 4.8(e), however with two links $l = 0$, 1 incident to a same node in $\mathcal{B}^*$, i.e. $p_0^* = p_1^*$.

$$\begin{aligned} d_l^2 &= \|p - p_l\|^2 \qquad (\text{with } l = 0 \text{ or } 1 \text{ and } p_0^* = p_1^* = \mathbf{0}) \\ 0 &= \begin{bmatrix} A_2 + B_2 & 2\,C_2 & A_2 - B_2 \end{bmatrix} \begin{bmatrix} t^0 & t^1 & t^2 \end{bmatrix}^{\mathrm{T}} \end{aligned}$$

The first equation corresponds to these two links with $p_0^* = p_1^* = \mathbf{0}$, which can be realized by moving the original point of body $\mathcal{B}^*$ to this shared sensor. The first equation has 2 roots for p, since two circles has 2 intersection points. The second equation directly follows Eq. (4.5), by which and a known p, we can derive 2 roots for rotation R. Considering the reflection of body $\mathcal{B}^*$, $[p, R]$ can have 8 ambiguities. □

Theorem 4. *There are at most twelve ambiguities during the merging of body $\mathcal{B}^*$ into body $\mathcal{B}$, if*

- *they can satisfy the condition of DOC $\geq$ DOF, and*
- *they are connected by three links as shown in Fig. 4.8(e).*

Proof. The following derivations assumes that body $\mathcal{B}$ contains at least three nodes and the number of links n equals 3, to make it a determined system with DOC = DOF which is shown in Fig. 4.8(e). It is also assumed that there are no zero-length links $\forall\, l \in [0,\, n),\ d_l \neq 0$, and there do not exist two links sharing a sensor $\forall\, l_1\, l_2 \in [0,\, n),\ l_1 \neq l_2 \rightarrow p^*_{l_1} \neq p^*_{l_2}$.

Before the elimination of variable t in Eq. (4.5), note that B_l, C_l are linear in p, and A_l is quadratic in p. The next step therefore is to reduce the degree of A_l by eliminating the quadratic term $\|p\|^2$; otherwise, the resultant, after the elimination variable t in Eq. (4.5), can have a high degree in p.

Firstly, it is assumed that in Eq. (4.4) p^*_0 equals zero, which can be realized by a temporary translation of body $\mathcal{B}^*$'s coordinate frame, and thus Eq. (4.6) can be established.

$$d_0^2 \quad = \quad \|p - p_0\|^2 \quad = \quad \|p\|^2 - 2\, p_0 \cdot p + \|p_0\|^2 \tag{4.6}$$

Therefore, the degree of A_l $(l = 1,\, 2)$ can be reduced to one in Eq. (4.7), by substituting $\|p\|^2$ with $d_0^2 + 2\, p_0 \cdot p - \|p_0\|^2$.

$$\begin{aligned} \mathbf{0} &= \begin{bmatrix} A_1 + B_1 & 2\, C_1 & A_1 - B_1 \\ A_2 + B_2 & 2\, C_2 & A_2 - B_2 \end{bmatrix} \begin{bmatrix} t^0 & t^1 & t^2 \end{bmatrix}^{\mathrm{T}} \\ A_l &= -2\,(p_l - p_0) \cdot p + (\|p_l\|^2 - \|p_0\|^2) + \|p^*_l\|^2 - (d_l^2 - d_0^2) \\ B_l &= 2\, p^*_l \cdot p - 2\, p^*_l \cdot p_l \qquad\qquad C_l \ = \ 2\, \vec{p}^{\,*}_l \cdot p - 2\, \vec{p}^{\,*}_l \cdot p_l \end{aligned} \tag{4.7}$$

By elimination of variable t in Eq. (4.7), Eq. (4.8) is established. The elimination technique used is linear elimination since the sufficient and necessary condition for

the two polynomials in Eq. (4.7) to have a common root for variable t is that the determinant of their associated Sylvester matrix M should vanish.

$$\begin{aligned} 0 &= \det(M) \\ M &= \begin{bmatrix} A_1+B_1 & 2\,C_1 & A_1-B_1 & 0 \\ 0 & A_1+B_1 & 2\,C_1 & A_1-B_1 \\ A_2+B_2 & 2\,C_2 & A_2-B_2 & 0 \\ 0 & A_2+B_2 & 2\,C_2 & A_2-B_2 \end{bmatrix} \end{aligned} \tag{4.8}$$

This common root, by eliminating t's quadratic term, is

$$t = \frac{(A_1-B_1)\,(A_2+B_2) - (A_1+B_1)\,(A_2-B_2)}{2\,C_1\,(A_2-B_2) - (A_1-B_1)\,2\,C_2}\,. \tag{4.9}$$

The polynomial without t in Eq. (4.8) is quartic in p, because each row of M is linear in p. We however can further reduce Eq. (4.8) to a cubic polynomial by substitutions of $\|p\|^2$ with $d_0^2 + 2\,p_0 \cdot p - \|p_0\|^2$. Eq. (4.8) can be expanded to Eq. (4.10).

$$\begin{aligned} 0 &= (B_1\,C_2 - C_1\,B_2)^2 + 2\,A_1\,A_2\,(B_1\,B_2 + C_1\,C_2) \\ &\quad - A_1^2\,(B_2^2+C_2^2) - A_2^2\,(B_1^2+C_1^2) \end{aligned} \tag{4.10}$$

The following analysis shows that Eq. (4.10) is cubic in p. With

$(p_1^* \cdot p)\,(\vec{p}_2^* \cdot p) - (\vec{p}_1^* \cdot p)\,(p_2^* \cdot p) = (p_1^* \cdot \vec{p}_2^*)\,\|p\|^2$, we have

$$\begin{aligned}(B_1\,C_2 - C_1\,B_2)\,/\,4 \quad &= \quad (p_1^* \cdot \vec{p}_2^*)\|p\|^2 \\ &- (p_1^* \cdot p)(\vec{p}_2^* \cdot p_2) - (p_1^* \cdot p_1)(\vec{p}_2^* \cdot p) + (p_1^* \cdot p_1)(\vec{p}_2^* \cdot p_2) \\ &+ (\vec{p}_1^* \cdot p)(p_2^* \cdot p_2) + (\vec{p}_1^* \cdot p_1)(p_2^* \cdot p) - (\vec{p}_1^* \cdot p_1)(p_2^* \cdot p_2)\,.\end{aligned}$$

Moreover, we have $(p_l^* \cdot p)^2 + (\vec{p}_l^* \cdot p)^2 = \|p_l^*\|^2\|p\|^2$ and
$(p_l^* \cdot p)\,(p_l^* \cdot p_l) + (\vec{p}_l^* \cdot p)\,(\vec{p}_l^* \cdot p_l) = \|p_l^*\|^2\,(p_l \cdot p)$. Therefore,

$$(B_l^2 + C_l^2)\,/\,4 \quad = \quad \|p_l^*\|^2\,(\|p\|^2 - 2\,p_l \cdot p + \|p_l\|^2)\,.$$

Because $(p_1^* \cdot p)\,(p_2^* \cdot p) + (\vec{p}_1^* \cdot p)\,(\vec{p}_2^* \cdot p) = (p_1^* \cdot p_2^*)\,\|p\|^2$,

$$\begin{aligned}(B_1\,B_2 + C_1\,C_2)\,/\,4 \quad &= \quad (p_1^* \cdot p_2^*)\,\|p\|^2 \\ &- (p_1^* \cdot p\ )(p_2^* \cdot p_2) - (p_1^* \cdot p_1)(p_2^* \cdot p\ ) + (p_1^* \cdot p_1)(p_2^* \cdot p_2) \\ &- (\vec{p}_1^* \cdot p_1)(\vec{p}_2^* \cdot p\ ) - (\vec{p}_1^* \cdot p\ )(\vec{p}_2^* \cdot p_2) + (\vec{p}_1^* \cdot p_1)(\vec{p}_2^* \cdot p_2)\,.\end{aligned}$$

By the substitution of $\|p\|^2$ with $d_0^2 + 2\,p_0 \cdot p - \|p_0\|^2$, $B_1C_2 - C_1B_2$, $B_l^2 + C_l^2$, $B_1B_2 + C_1C_2$ can be reduced to be linear in p, and thus reduce Eq. (4.10) to be cubic in p.

The quadratic curve about p in Eq. (4.6) and the cubic curve about p in Eq. (4.10), according to Bézout's theorem, have $2 \times 3 = 6$ intersection points in complex space. In fact, given special inputs, these 6 intersection points can all stay in real space, according to our simulation results. Given a translation p, there is only one t value as indicated by Eq. (4.9) and thus only one rotation R with $t = \tan\frac{\gamma}{2}$. Considering the reflection of body $\mathcal{B}^*$, the number of solutions for $[p,\, R]$ is at most 12. □

4.5.2 Flip Ambiguity Enumeration

The key point of our inflexible body merging is the ability to enumerate finite ambiguities during the merging of two bodies. The existence of finite ambiguities can be caused by

- the lack of redundancy in constraints as shown in Fig. 4.8,
- or the collinearity of constraints as shown in Fig. 4.9.

The previous subsection has presented the method to enumerate ambiguities by solving polynomial equation system. However, such a symbolic method cannot enumerate the flip ambiguities in Fig. 4.9 where the constraint point set can be roughly collinear due to the interferences of ranging noise.

In this subsection, we propose a numerical algorithm to enumerate flip ambiguities during body merging, with the presence of ranging noise. Our observation is that the ambiguities in Fig. 4.8(a-d) and in Fig. 4.9(a-e) have the symmetry that if flipping one ambiguity across the depicted lines, we can get the other ambiguity or ambiguities. For example, in Fig. 4.9(d), a second ambiguity can be obtained by flipping the depicted ambiguity across the line through nodes $1, 2^*, 3^*$. The difficulty is that when enumerating flip ambiguities for the case in Fig. 4.8(d), there are two flip lines ϕ_0, ϕ_1 and four flip ambiguities. The second ambiguity is obtained by flipping the first ambiguity across ϕ_0, the third ambiguity is obtained by flipping the first ambiguity across ϕ_1, and the fourth ambiguity is obtained by flipping the first ambiguity across ϕ_0 then across ϕ_1.

We present the pseudo code of our flip ambiguity enumeration algorithm in Algorithm 6, which provides good localization robustness by testing the collinearity of

each constraint point set $P_{\mathcal{C}}(k)$, as defined in Theorem 1 and described at Line 3-5. This algorithm also has the advantage of enumerating flip ambiguities with the presence of ranging noise, thanks to the ability of Algorithm 5 (invoked at Line 7 of Algorithm 6) to detect the rough collinearity of a constraint point set.

Algorithm 6: EnumerateFlipAmbiguities

Input: An ambiguity of body $\mathcal{B}$ (call it body $\mathcal{B}$ for short);
An ambiguity of body $\mathcal{B}^*$ (call it body $\mathcal{B}^*$ for short);
Constraint Set $\mathcal{C} = \{\ell_l\} = \{\,[n_l,\, n_l^*,\, d_l,\, w_l]\,\}$
Output: All flip ambiguities when merging two ambiguities $\mathcal{B}$,$\mathcal{B}^*$

1 **begin**
  `// Step 1. obtain the first ambiguity`
2 Get the first ambiguity by invoking Algorithm 4
  `// Step 2. obtain the other ambiguities`
3 **switch** *number of nodes in body* $\mathcal{B}^*$ **do** `// Theorem 1`
4 **case** 1*:* $k = 2^{|\mathcal{C}|}-1$; **case** 2: $k=1$; **otherwise**: $k=0$
5 **for** $k \mathrel{+}= 2^{|\mathcal{C}|} - 2^{|\mathcal{C}|-NumOfSharedNodes}$; $k < 2^{|\mathcal{C}|}$; $k{+}{+}$ **do**
6 **if** *the nodes in constraint point set* $P_{\mathcal{C}}(k)$ *coincide at one position* **then** **return** $\emptyset$ to indicate the failure
7 **if** *constraint point set* $P_{\mathcal{C}}(k)$ *is collinear around a line* ϕ *(as tested by Algorithm 5)* **then** record the line ϕ
8 **if** *a flip line* ϕ_0 *has been found* **then** get the second ambiguity by flipping the first ambiguity across line ϕ_0
9 **if** *a second flip line* ϕ_1 *has been found* **then** get the third ambiguity by flipping the first ambiguity across ϕ_1; get the fourth ambiguity by flipping the first one across ϕ_0 and ϕ_1

4.6 Simulation Results

This section firstly compares our optimal body merging algorithm in Algorithm 3 with the traditional noise toleration algorithms (i.e. Multilateration [Foy, 1976] and Patch Stitching [Horn et al., 1988]) to verify that our algorithm is more generalized than the traditional algorithms in tolerate ranging noise during body merging. Then

we show that our flip ambiguity enumeration algorithm in Algorithm 6 can improve the robustness of body merging. We further compare our IIBM algorithm in Algorithm 2 with state-of-the-art CALL [Wang et al., 2008], to verify that our algorithm can achieve higher localization percentage than CALL. Finally, our IIBM algorithm are applied to localize sparse networks deployed in either convex or concave regions.

4.6.1 Simulation Settings

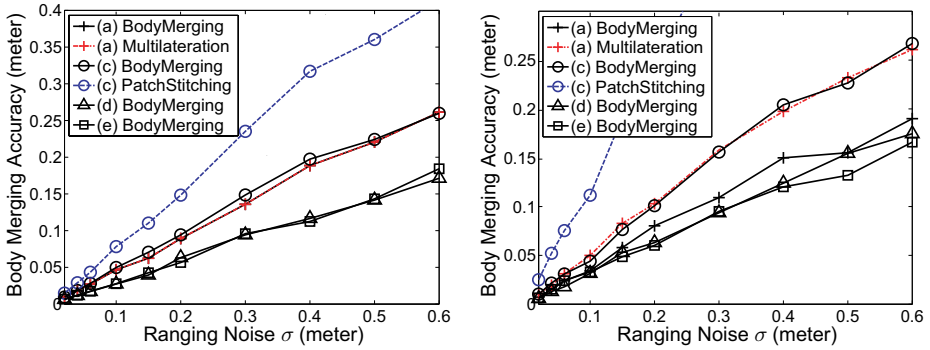
The adopted ranging system is ultrasonic TOA with 5m ranging radius and 4 cm ranging noise σ, which is also used in [Moore et al., 2004]. Our simulations evaluate the proposed localization algorithms by the metrics listed in Table 4.1. The adjustable system parameters are also listed there and we investigate their impact on the evaluated metrics.

Table 4.1: Evaluated Metrics and Adjusted System Parameters

LocError	Localization Error of a node is the distance between its true location and its unique location estimate.
GCF Anchor LocError	Localization Error of a GCF anchor is the distance between its true location and its declared location.
Body Merging Accuracy	Body Merging Accuracy is the mean-squared error of all constraints between two bodies, as formalized in Eq. (4.2), which is also used by [Goldenberg et al., 2006] since mean-squared error of constraints is proportionate to average LocError by Cramér-Rao bound if ranging noise is Gaussian noise.
LocPercentage	Localization Percentage is the percentage of nodes that can be localized finitely, among all network nodes.
NtwkDegree	Network Degree is average number of neighbors of a sensor, to which the distances can be measured.
AncPercentage	Anchor Percentage is the percentage of GCF anchors deployed to the network, among all network nodes.
NtwkRegion	Network Region is the region to deploy the network, which can be rectangular, O-Shaped, U-Shaped, etc.

4.6.2 Body Merging Accuracy By Tolerating Ranging Noise

This simulation is to show that our optimal body merging algorithm in Algorithm 3 can achieve high body merging accuracy for all the cases in Fig. 4.7(a-e). Our algorithm is thus more generalized than the traditional noise toleration algorithms, i.e. multilateration [Foy, 1976] and patch stitching [Horn et al., 1988]. Multilateration only handles the case in Fig. 4.7(a). Patch stitching is designed for the case in Fig. 4.7(c).



(a) GCF Anchor LocError = 0 (b) GCF Anchor LocError = σ

Fig. 4.12: Comparisons in Body Merging Accuracy.

Fig. 4.12 depicts the relation between ranging noise σ and body merging accuracy (check Table 4.1) during the merging of a local body into the global body. Fig. 4.12(a) assumes that GCF anchor localization error is zero, and Fig. 4.12(b) assumes that GCF anchor LocError is σ to emulate the effect of error accumulation, i.e. the increase of localization error of nodes in the global body with the increase of number of nodes.

Fig. 4.12(a) shows that our optimal body merging algorithm can tolerate noise and improve body merging accuracy for all the cases in Fig. 4.7 in a unified manner. This major advantage is because we adopt the physical model of rigid body dynamics to align two groups of nodes. Fig. 4.12(a) also shows the accuracy of our body

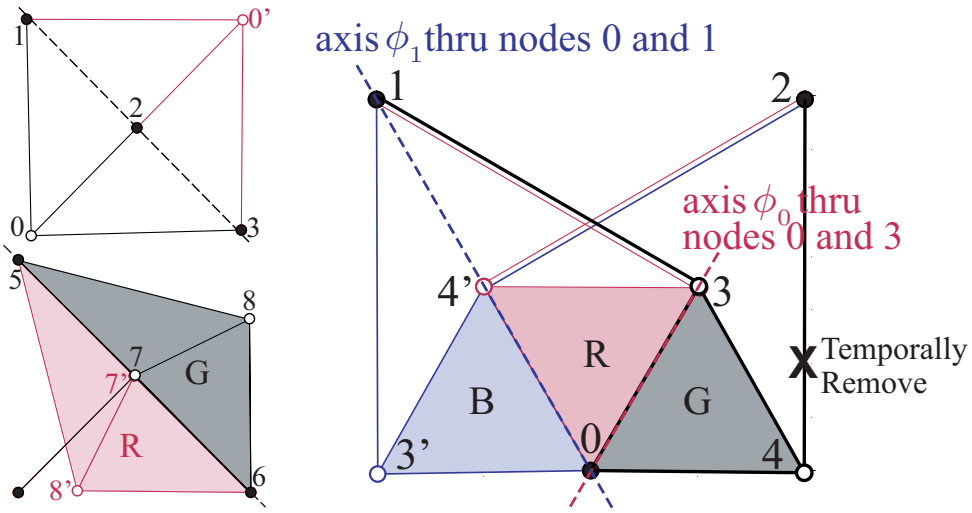
merging algorithm is comparable with multilateration for case (a). Our body merging algorithm can outperform patch stitching for case (c) because our body merging algorithm refines the merged body $\mathcal{B}^+$ (see Algorithm 4) to tolerate the ranging noise in the local body $\mathcal{B}^*$.

Fig. 4.12(b) shows that our optimal body merging algorithm can alleviate the error accumulation problem, i.e. the increased uncertainty in node positions within a body when the body contains more nodes. To simulate such an effect, the GCF anchor LocError is configured to σ in Fig. 4.12(b). In this situation, our optimal body merging algorithm outperforms multilateration during case (a), and outperforms patch stitching during case (c). Such an advantage is because we apply refinement to merged body $\mathcal{B}^+$ (see Algorithm 4) to tolerate the positioning uncertainties in the global body $\mathcal{B}$.

4.6.3 Robust Body Merging against Unexpected Flipping

During body merging, it is necessary to discover the collinear geometry in constraint set $\mathcal{C}$ and enumerate flip ambiguities (see Section 4.4.2 and Section 4.5.2). The simulation in Fig. 4.13 show that our algorithm in Algorithm 6 can enumerate flip ambiguities for the cases in Fig. 4.9(a)(c)(d), and can thus improve the robustness of inflexible body merging.

Fig. 4.13(a) shows that Algorithm 6 can enumerate flip ambiguities for sensor merging in Fig. 4.9(a) and patch stitching in Fig. 4.9(c). Sensor 0 has two ambiguous positions since anchors $1 \sim 3$ are collinear. The patch with nodes $5 \sim 8$ contains two anchors 5, 6 and has one link [4, 7] connecting it to anchor 4. This patch has two ambiguities (with gray and red colors) since the three nodes $5 \sim 7$ are collinear.



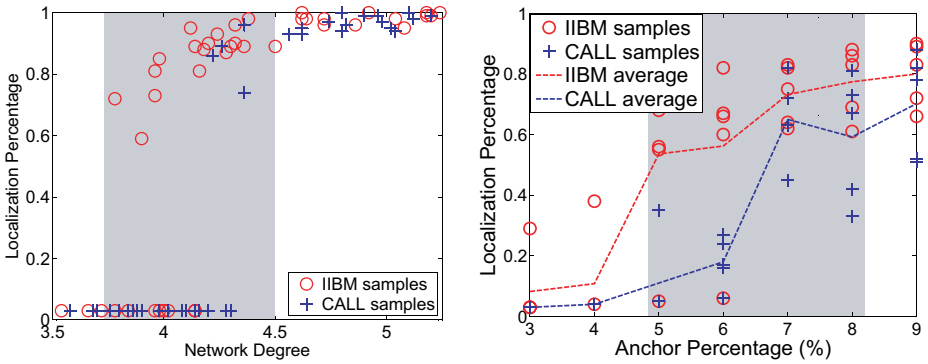
(a) Test Cases (a)(c). (b) Test Case (d).

Fig. 4.13: Enumerate Flip Ambiguities During Body Merging in Cases (a)(c)(d) of Fig. 4.9 due to Constraints Collinearity.

Fig. 4.13(b) shows that Algorithm 6 can enumerate flip ambiguities for patch merging in Fig. 4.9(d). The patch with nodes $0, 3, 4$ has an redundant constraint set that contains one anchor node 0 and two links [1, 3], [2, 4]. This patch has three ambiguities with gray, red and blue colors. To enumerate the three ambiguities, the constraint set is firstly reduced to be determined (i.e. DOC = DOF) by removing a link, e.g. [2, 4]. Then the first ambiguity can be obtained (e.g. with the gray color) together with the two illustrated flip axes ϕ_0 and ϕ_1. The red ambiguity is obtained, by flipping the gray ambiguity across axis ϕ_0. The blue ambiguity is obtained, by flipping the gray ambiguity firstly across ϕ_0 then across ϕ_1.

4.6.4 Localization Percentage in Sparse Networks

This simulation is to verify that our IIBM algorithm in Algorithm 2 can achieve high localization percentage in sparse networks with low connectivity and sparse anchor distribution. We adjust network degree and anchor percentage to investigate their impacts on localization percentage and show that we can achieve higher localization percentage than CALL [Wang et al., 2008].



(a) Fixed Anchor Number = 3 (b) Fixed Network Degree ≈ 3.8

Fig. 4.14: Comparisons in Localization Percentage.

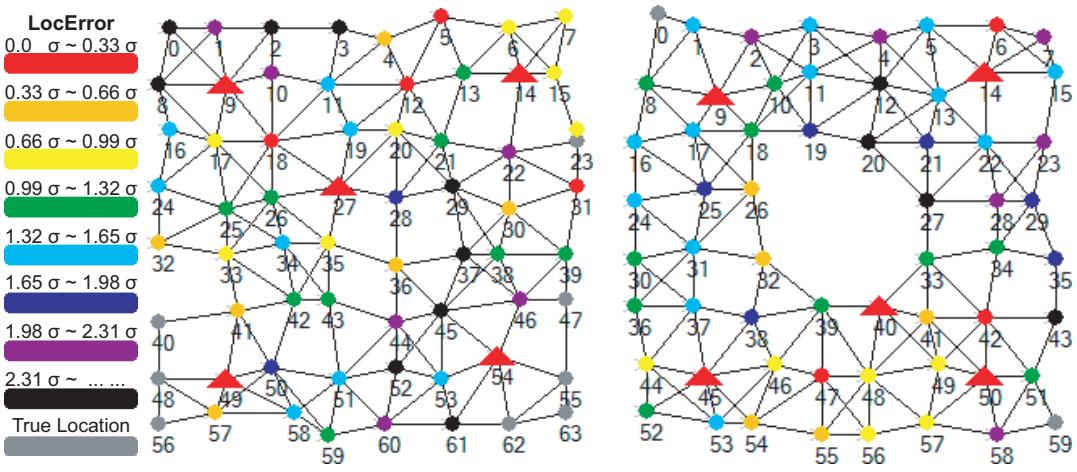
Fig. 4.14(a) depicts the relation between network degree and anchor percentage. Fig. 4.14(a) shows that the localization percentage of both IIBM algorithm and CALL approaches 1 when network degree is larger than 4.5. But when network degree is between 3.7 and 4.5, IIBM algorithm can outperform CALL by roughly two times (see Fig. 4.14(b)). This is because CALL does not consider ambiguous cases (c)(d) in Fig. 4.8 for component merging, which would reduce the chance for local bodies to expand. In contrast, IIBM algorithm can handle all ambiguous body merging cases (a-e) and apply these cases equally to the merging and embedding of local bodies.

Fig. 4.14(b) depicts the relation between anchor percentage and localization percentage. Fig. 4.14(b) shows that IIBM algorithm strongly outperforms CALL when

anchor percentage is between 5% and 8%. This is because IIBM algorithm gives local bodies greater chance to expand, and a larger local body has better chance to be localized in GCF.

4.6.5 Performance in Convex and Concave Networks

Our IIBM algorithm in Algorithm 2 is applied to sparse networks deployed in concave and convex regions (see in Fig. 4.15) to show our algorithm is accurate and robust in both regions. In Fig. 4.15, GCF anchors are drawn as triangles and sensors are drawn as dots. For each sensor, if it has unique location estimate, a colored dot is used to to show its estimated location and the color used corresponds to its localization error (see the legend on the left of Fig. 4.15(a)). The nodes that can not be uniquely localized are shown as gray dots at their true locations (e.g. nodes $55, 62, 63$ in Fig. 4.15(a)).



(a) Convex Rectangular Field. (b) Concave O-Shaped Field.

Fig. 4.15: Localization by IIBM algorithm. (a) NtwkDegree = 4.81, LocError = $1.58\,\sigma$, LocPercentage = 85%. (b) NtwkDegree = 5.13, LocError = $1.16\,\sigma$, LocPercentage = 96.7%.

In both the convex topology in Fig. 4.15(a) and the concave topology in Fig. 4.15(b), the average localization error of localized nodes is kept below $2\,\sigma$. This good accuracy is because (1) we can provide good body merging accuracy by our optimal body merging algorithm as shown in Fig. 4.12(b), and (2) we can ensure body merging robustness by enumerating the flip ambiguities. For example, nodes $40, 48, 56$ cannot be localized uniquely due to the collinearity of nodes $41, 49, 57$. Moreover, for the two sparse networks in Fig. 4.15, the localization percentage is well kept above 80%, which is consistent with the results in Fig. 4.14.

4.7 Conclusion

This chapter focused on fine-grained iterative localization of wireless sensor networks. We have proposed an optimal body merging algorithm that can tolerate ranging noise when aligning two bodies. This algorithm can handle all body merging cases in a unified manner and the simulation results show that it is more accurate than the traditional multilateration and patch stitching by mitigating error accumulation. We also proposed a flip ambiguity enumeration algorithm that can improve the localization robustness by discovering rough constraint collinearity with the presence of ranging noise. It is finally verified that our network localization algorithm called IIBM algorithm can achieve higher localization percentage than state-of-the-art CALL and our demonstration shows that it can work well in both convex and concave networks.

Chapter 5

Range-Based Localization of Wireless Sensor Networks: Robustness Against Outliers

5.1 Overview

For wireless sensor networks, a wide range of applications and protocols require sensor nodes to know their locations. For example, environmental monitoring application needs data reported by sensor nodes to be geographically meaningful, and routing protocol can reduce its communication cost by geographical routing. Therefore, a plethora of previous studies can be found to automatically derive sensor locations, which is known as *network localization* [Eren et al., 2004, Goldenberg et al., 2006, Horn et al., 1988, Li and Liu, 2007, Lim and C., 2005, Moore et al., 2004, Niculescu and Nath, 2003, Priyantha et al., 2003, Savvides et al., 2003b, Shang and Ruml, 2004, Wang et al., 2008, Whitehouse et al., 2005]. These studies generally assume

the distances between neighboring nodes in a topology can be accurately measured by *ranging techniques*, e.g. ultrasonic TOA (Time-Of-Arrival) and ultra-wideband RF TOA.

For the measuring of inter-node distances, it is inevitable to have erroneous distance measurements that deviate significantly from true distances (call them *outlier distances* or outlier links). The presence of these outlier distances can be caused by malfunctions of ranging hardware, severe natural interferences to ranging signals, or malicious attacks. For example, ultrasonic TOA may generate outlier distances with enlarged estimates due to non-line-of-sight propagation of sound signals [Whitehouse et al., 2005].

These inevitable outlier distances can severely degrade the accuracy of network localization algorithms. For example, in Fig. 5.1, nodes 1, 2, 3, 4 are anchors nodes whose locations are already known. The sensor 0 can measure distances to them. Each distance measurement is drawn as a dashed circle centered at the corresponding anchor node. Among the four links, the link [4, 0] is an outlier whose measurement is much smaller than the true value. As misled by this outlier link, location estimate of sensor 0 converges to the severely inaccurate position 0′ following the red polyline, if applying multilateration.

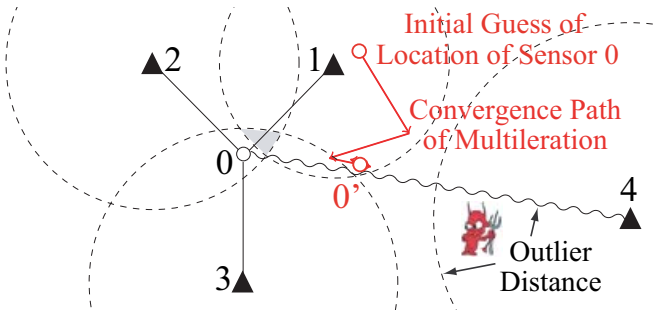


Fig. 5.1: Impact of Outliers on Multilateration Accuracy.

These harmful outlier distances need to be identified and rejected during network localization, which is called *robust network localization*. Previous studies in this field are focused on adding outlier rejection ability to multilateration (called *robust multilateration* [Kiyavash and Koushanfar, 2007, Kung et al., 2009, Li et al., 2005, Liu et al., 2005, Wang et al., 2007]), because multilateration is a basic operation that can be applied *iteratively* to localize a network. Here "iteratively" means a node can be localized if it can measure distances to at least three anchors, and a node after being localized can be upgraded to an anchor [Savvides et al., 2003b].

Iterative multilateration becomes weak and even powerless for localization in sparse networks where the node degree can be six or less. Thus, we must resort to another basic operation called *patch merging* where a patch is a group of nodes forming a rigid structure and two patches can be merged to generate a larger patch if they have enough connectivity to confine their relative motions [Horn et al., 1988, Moore et al., 2004, Shang and Ruml, 2004, Wang et al., 2008]. Thanks to patch merging, the percentage of localizable nodes can be increased by at least 30% in sparse networks with the average degree between 5 and 7 (see the simulations in Section 5.6.3).

However, the accuracy of this useful patch merging operation is still vulnerable to outlier distances, due to its nature of least square estimator. There is no previous work to invent *robust patch merging* that can reject outliers during patch merging. Thus, we propose such a robust operation as the basis of robust network localization. This operation is more generalized and powerful than robust multilateration since it can reject outliers either when merging a single node with a patch (i.e. multilateration) or when merging two patches (i.e. patch merging). To implement such generalized

operation, our basic idea is to find a *consistent* subset of the links connecting two patches. We say a subset of links is "consistent" if it has no detectable outliers (i.e. the residue of each link in this subset is below a threshold proportionate to ranging noise).

Robust patch merging does not directly imply the robust network localization. This is because the robust patch merging requires *sufficient redundancy* in connectivity between two patches to reject outlier links. This redundancy requirement however may not be satisfied in sparse subregions of a network. An outlier link in such a sparse region thus may be non-rejectable and skew the final location estimates. We propose to make the robust patch merging operation explicitly report the existence of *non-rejectable outliers*. When receiving such a report, a network localization algorithm can isolate the non-rejectable outliers. This network localization algorithm is called *RobustLoc* which according to our proof can reject outlier links reliably. Compared with robust multilateration [Kiyavash and Koushanfar, 2007, Kung et al., 2009, Li et al., 2005, Liu et al., 2005, Wang et al., 2007] that can easily get stuck in sparse networks, RobustLoc can reliably reject outlier links in sparse networks. Our simulation results show that in sparse networks with 5.5 degree, RobustLoc can localize 90% nodes, which strongly outperforms robust multilateration methods.

Besides the toleration of outlier links, another key issue is the toleration of *outlier anchor nodes*. Anchor nodes constitute only a small proportion of network nodes in a network topology. These anchor nodes have prior knowledge of their locations in a Global Coordinate Frame (GCF), e.g. in the GPS coordinate frame which is meaningful to most of potential users. However, it is inevitable to have outlier anchor

nodes declaring erroneous locations that significantly deviate from their true locations. The sources of outlier anchors can be misconfigurations or malicious attacks, e.g. Replay attack (i.e. attacker overhears an anchor location declaration and replays this declaration at other places) and Sybil attack (i.e. attacker compromises an anchor node, exploits its identity and makes erroneous anchor declarations at different places [Newsome et al., 2004]). Outlier anchors are more harmful than outlier links because multiple outlier anchors may *collude* and declare positions in the same coordinate frame which is different from the GCF.

We find that *outlier anchors toleration* is different from outlier links toleration for two reasons: (1) falsely rejecting normal links incident to outlier anchors may cause troubles which will be detailed by simulations in Section 5.5.1; (2) different from outlier links, outlier anchors may collude due to malicious attacks. Thus the rejection of colluding outlier anchors needs global knowledge to exploit the *majority of benign anchors*, i.e. benign anchors outnumber outlier anchors in the whole network. Therefore, we propose an enhancement to RobustLoc algorithm that firstly constructs a largest local patch by an iterative local patch merging process excluding all the anchor declarations. Then by merging this largest local patch with the global patch containing all anchor declarations, we can reject colluding outlier anchors by *voting*. Such voting is implemented by reusing our robust patch merging operation, which reflects an elegant design.

As a summary, this chapter makes the following contributions. (1) We propose the robust patch merging operation which is more generalized and powerful than robust multilateration [Kiyavash and Koushanfar, 2007, Liu et al., 2005, Wang et al.,

2007]. Our RobustLoc algorithm thus can effectively reject outlier links and meanwhile achieve high localization percentage in sparse networks. In contrast, robust multilateration method can only localize a small proportion of nodes in sparse networks. (2) Our robust patch merging operation handles the non-rejectable outlier distances due to insufficient connectivity in sparse subregions, which is neglected before. (3) Our RobustLoc algorithm can reject both outlier links and outlier anchors. In contrast, a recent chapter [Jian et al., 2010] can only reject outlier links, based on the enumeration of realizable generic cycles. (4) Our RobustLoc algorithm can tolerate multiple outlier anchors which may collude under malicious attacks. These declared contributions will be validated by high-fidelity simulations with practical system parameters from [Moore et al., 2004, Savvides et al., 2003b].

The rest of this chapter is organized as follows. Section 5.2 introduces the two basic operations of network localization, i.e. multilateration and patch merging. Section 5.3 presents our robust patch merging that can reject outlier links for these two basic operations. Section 5.4 presents how we enhance network localization algorithm to make it robust against outlier links. Section 5.5 addresses the problem as how to reject outlier anchors. Section 5.6 shows simulation results. Section 5.7 concludes the whole chapter.

5.2 Two Basic Operations for Network Localization Problem

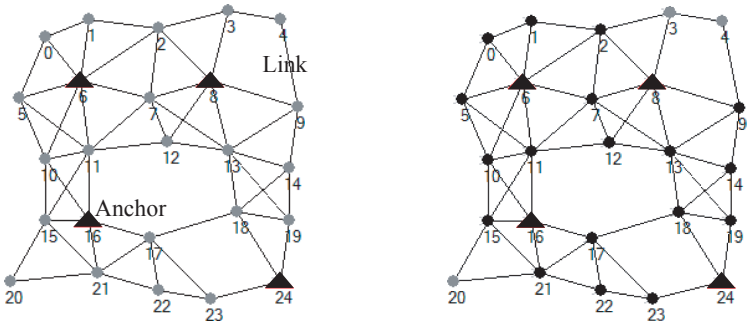
In this section, we firstly introduce the sensor network localization problem and then introduce the two basic operations to solve such a problem, i.e.

1. multilateration that merges a sensor with a patch,

2. patch merging that merges two patches.

Here, a patch is a group of nodes forming a rigid structure without continuous deformation.

Network Localization Problem. To localize a network, the required inputs are a set of anchors, whose locations are already known, and a set of link length measurements as shown in Fig. 5.2(a). The outputs of a network localization algorithm are the location estimates of the nodes that can be uniquely localized. These location estimates are depicted in Fig. 5.2(b) as black dots. The nodes that have ambiguous location assignments do not have black dots in Fig. 5.2(b). For example, node 20 has ambiguous location estimates since it can flip across the line through nodes $15, 21$.



(a) Input: Anchors and Links (b) Output: Location Estimates

Fig. 5.2: Input and Output of Network Localization Problem.

To solve this network localization problem, researchers try to find out how a group of nodes can be inter-connected to form a rigid structure which is free from any deformations (i.e. either flip or flex). Such a structure can be localized if it contains three non-collinear anchors to fix it on a plane. Such a structure is named a

generically globally rigid graph [Eren et al., 2004] (or globally rigid graph for short) and is defined in Theorem 5.

Theorem 5 (Globally Rigidity & Generically Rigidity)**.** *Graph G with at least four vertices is globally rigid in $\mathbb{R}^2$ if G is 3-connected and redundantly rigid. Graph G is* redundantly rigid *if G is generically rigid upon the removal of any edge. Graph G is generically rigid if it can satisfy Laman's Theorem. Please refer to [Eren et al., 2004] for details and proofs.*

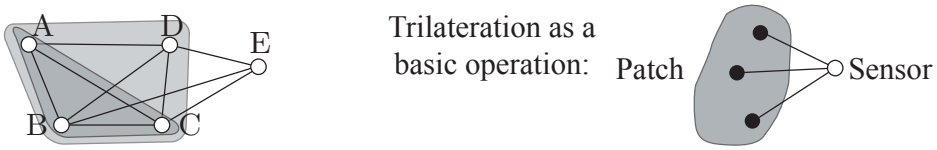


Fig. 5.3: Patch Expansion by Iterative Trilateration.

However, it is NP-hard to find in a network topology all the globally rigid subgraphs and test whether they can be localized [Eren et al., 2004]. Thus people construct globally rigid patches by merging nodes into patches iteratively, which is called *iterative trilateration* with polynomial computational cost [Savvides et al., 2003b]. For example, in Fig. 5.3, we start from the simplest rigid structures, i.e. triangle $\{A, B, C\}$. We expand this rigid structure (called a patch) by merging node D into it, since there are three links connecting this node with the patch $\{A, B, C\}$. The operation that merges a node into a patch is called trilateration. By applying trilateration iteratively, we can keep adding new nodes into the patch. The patches expanded in this way are called *trilateration graphs* which are guaranteed to be globally rigid [Eren et al., 2004].

However, iterative trilateration can easily get stuck in sparse networks with low connectivity and with sparse anchor distribution, e.g. the network in Fig. 5.2(a). The

recent work in [Horn et al., 1988, Wang et al., 2008] thus considers not only merging a node into a patch but also merging two patches which is depicted in Fig. 5.4(b-e) and described in Definition 6.

Definition 6 (Globally Rigid Patch Merging [Wang et al., 2008])**.** *Left patch G_L and right patch G_R can be merged together to generate a larger resultant patch. This merging is globally rigid merging if the two patches G_L and G_R can fulfill one of the following conditions with*

- *three shared nodes in Fig. 5.4(b).*
- *two shared nodes and one connecting link in Fig. 5.4(c);*
- *one shared node and two connecting links in Fig. 5.4(d);*
- *at least four connecting links in Fig. 5.4(e).*

Moreover, for all these shared nodes and links, they should be relevant to at least three nodes in left patch G_L, and relevant to at least three nodes in right patch G_R. We call it "globally rigid merging", because the resultant patch is globally rigid if G_L and G_R are both globally rigid.

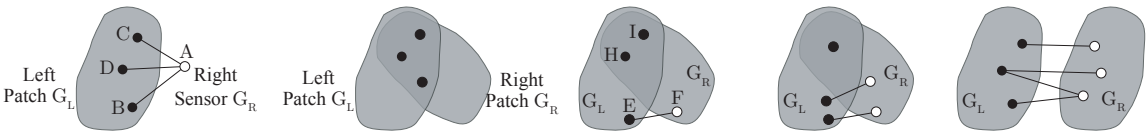


Fig. 5.4: Globally rigid merging cases. (a) merges patch G_L with sensor G_R. (b-e) merge two patches G_L,G_R.

With patch merging, we can discover more globally rigid subgraphs in sparse networks than using trilateration alone. This is because for trilateration in Fig. 5.4(a)

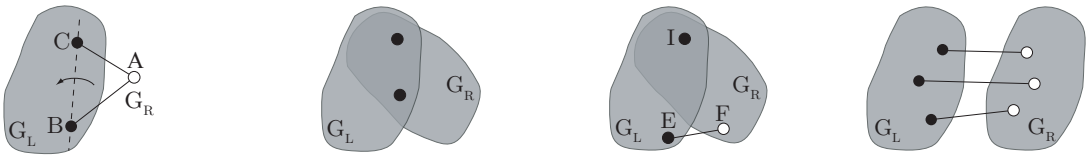
we can use only the one-hop information in node A's neighborhood, while for patch merging in Fig. 5.4(b-e) we can exploit multihop information to align two patches.

To further increase the ability of discovering globally rigid structures (e.g. wheel graphs), a patch can be constructed with finite realizations, which is called a generically rigid patch. For example, SWEEPS [Goldenberg et al., 2006] argues a sensor node can be merged into a patch with only two links connecting them. The node however have two possible realizations as shown in Fig. 5.5(a) by flipping across the line through nodes B, C. The patches expanded with finite realizations are generically rigid but not globally rigid. We thus call the merging "generically rigid merging". Two patches can also be merged with finite realizations [Wang et al., 2008], which is described in Definition 7 and illustrated in Fig. 5.5(b)(c)(d).

Definition 7 (Generically Rigid Patch Merging [Wang et al., 2008])**.** *Left patch G_L and right patch G_R can be merged together to generate a larger resultant patch. This merging is generically rigid merging if the two patches G_L and G_R can fulfill one of the following conditions with*

- *two shared nodes in Fig. 5.5(b).*
- *one shared nodes and one connecting link in Fig. 5.5(c);*
- *at least three connecting links in Fig. 5.5(d).*

Moreover, for all these shared nodes and links, they should be relevant to at least two nodes in left patch G_L, and relevant to at least two nodes in right patch G_R. We call it "generically rigid merging", because the resultant patch is generically rigid if G_L and G_R are both generically rigid.



(a) Merge a Sensor to a Patch (b) Merge Two Patches (c) Merge Two Patches (d) Merge Two Patches

Fig. 5.5: Generically rigid merging cases. (a) merges patch G_L with sensor G_R. (b-d) merge patches G_L, G_R.

Example. We can localize the network in Fig. 5.2(a) by merging patches iteratively. Triangle $\{2, 6, 7\}$ can be merged with triangle $\{6, 7, 11\}$ since they share two nodes, which corresponds to the case in Fig. 5.5(b). The resultant patch $\{2, 6, 7, 11\}$ is generically rigid, and this patch can be merged with triangle $\{7, 8, 12\}$ because they share node 7 and have two links $[2, 8]$ and $[11, 12]$. This globally rigid merging generates a patch $\{2, 6, 7, 8, 11, 12\}$ which is a six-node wheel graph and is globally rigid. This patch and the global patch (containing all the anchors $\{6, 8, 16, 18\}$) share two nodes $6, 8$ and has one link $[11, 16]$, which satisfies the globally rigid merging condition in Fig. 5.4(c). Thus, the nodes $2, 7, 11, 12$ can be merged into the global patch and be uniquely localized as shown in Fig. 5.2(b). Other black nodes can be localized similarly by iterative patch merging.

As a conclusion, for sensor network localization, there are two basic operations to expand patches: multilateration as shown in Fig. 5.4(a) & Fig. 5.5(a), and patch merging as shown in Fig. 5.4(b-e) & Fig. 5.5(b-d). Afterwards, we only use one term "patch merging", because multilateration can be regarded as a special case of patch merging if we regard an individual node (e.g. node A in Fig. 5.4(a)) as a special one-node patch. For patch merging, we use the following symbols all throughout this chapter.

G_L	left patch during patch merging
G_R	right patch (maybe just a sensor) during patch merging
L	set of links connecting G_L and G_R. Note that a node shared by G_L, G_R is contained in L as a zero-length link.

5.3 Robust Patch Merging Operation

This section presents robust patch merging that is still accurate in the presence of outlier links whose length measurements severely deviate from their real values. This robust patch merging is important since patch merging (with multilateration as a special case) is the basic operation for network localization. In Section 5.3.1, we discuss the detection of outlier links during patch merging. When outlier links are detected, we need to identify and remove these outliers, which is addressed in Section 5.3.2. Finally, Section 5.3.3 presents reliable outlier link rejection against collusion.

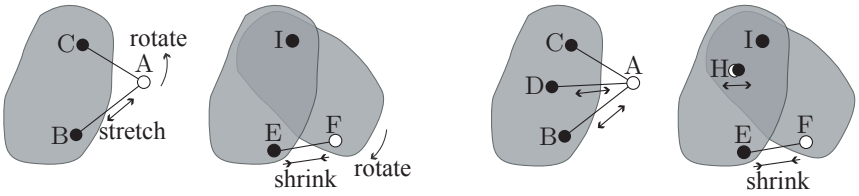
5.3.1 Outlier Link Detection during Patch Merging

We argue in Theorem 6 that outlier links can be detected if the merging is globally rigid as shown in Fig. 5.4. Such detection can be implemented as checking whether the links in set L have abnormally large deformations (or residues).

Theorem 6 (Outlier Detectability during Patch Merging)**.** *When G_L is merged with G_R, an outlier may exist in link set L. This outlier can be detected, if the merging is globally rigid as shown in Fig. 5.4. This outlier cannot be detected, if the merging is generically rigid but not globally rigid as shown in Fig. 5.5.*

Proof. If the merging is barely generically rigid, we cannot detect outliers since the removal of link ℓ ($\in L$) makes some part of the graph flexible. For example, in Fig. 5.6(a), if link $[A, B]$ is removed, node A can rotate around node C, and thus link

$[A, B]$ if put back can deform freely. In Fig. 5.6(a), link $[E, F]$ can also deform freely with the right patch rotating around node I.



(a) Non-Detectable Outlier Links for Generically Rigid Merging

(b) Detectable Outlier Links for Globally Rigid Merging

Fig. 5.6: Outlier Detectability of Basic Merging Operations.

Globally rigid merging upon the removal of any link is still generically rigid merging. Hence, for globally rigid merging, the deformation of outliers links incurs the deformation of normal links, which can be detected when we try to align G_L and G_R. For example, in Fig. 5.6(b), the stretch of link $[A, B]$ incurs the stretch of link $[A, D]$, which is detectable; the shrink of link $[E, F]$ incurs the stretch of link $[H, H]$, if regarding shared node H as a zero-length link. □

5.3.2 Outlier Link Rejection during Patch Merging

Upon the detection of outlier links, we need to identify which link or links are the outliers and then remove them. The basic idea is to find a subset of link set L that can let the two patches G_L and G_R merge without detectable outliers. This subset of links are trusted to be normal links, and the remaining links are identified as outlier links.

We present in Theorem 7 the necessary condition for outlier rejection during patch merging, i.e. there exists a subset of links that has no detectable outlier links and can satisfy the globally rigid merging condition for effective outlier detection.

Theorem 7 (Outlier Rejectability during Globally Rigid Patch Merging)**.** *When patch G_L is merged with patch G_R, outliers may exist in link set L. These outliers can be identified only if the merging of G_L and G_R is still globally rigid upon the removal of outliers in link set L.*

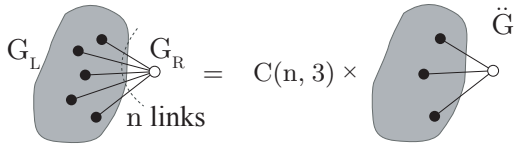
Proof. If the merging upon the removal of outliers becomes generically rigid merging, then this subgraph with outliers removed may not be outlier-free by Theorem 6. Thus we cannot tell whether we have rejected all the outliers. □

According to Theorem 7, we need to find an outlier-free subgraph of the original patch merging topology that is globally rigid. However, if we check each globally rigid subgraph on whether it contains outliers, the computational cost can be as high as $O(2^n)$, where n is the number of links connecting the two patches G_L and G_R. We thus only check each subgraph that is minimally globally rigid. This concept of *minimally globally rigid merging subgraph* is defined in Definition 8 and is illustrated in Fig. 5.7.

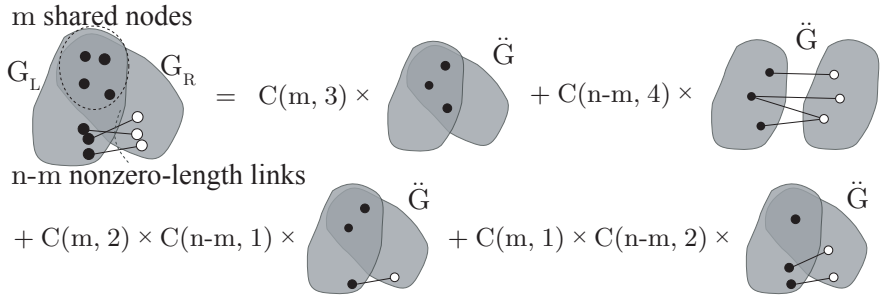
Definition 8 (Minimally Globally Rigid Merging Subgraph)**.** *Given a globally rigid patch merging topology G comprised of left patch G_L, right patch G_R and link set L with n links (see Fig. 5.7), $\ddot{G}$ is a minimally globally rigid merging subgraph of G, if $\ddot{G}$ can satisfy the following conditions:*

- *$\ddot{G}$ contains left patch G_L and right patch G_R;*
- *the links in $\ddot{G}$ that connects G_L,G_R is a subset of L that connects G_L,G_R in the original merging topology G;*
- *$\ddot{G}$ can satisfy globally rigid merging condition;*

- $\ddot{G}$ *cannot satisfy globally rigid merging condition upon the removal of any link connecting two patches* G_L,G_R.



(a) Enumerate minimally globally rigid merging subgraphs for multilateration



(b) Enumerate minimally globally rigid merging subgraphs for patch merging

Fig. 5.7: Enumerate Minimally Globally Rigid Merging Subgraphs for the Original Merging Topologies.

We illustrate in Fig. 5.7 the enumeration of *minimally globally rigid merging subgraphs* for multilateration and patch merging. Fig. 5.7(a) depicts the subgraph enumeration for multilateration, and there are $C(n, 3)$ such subgraphs, where n is the number of links. The studies in the field of robust multilateration [Kiyavash and Koushanfar, 2007, Wang et al., 2007] essentially adopt the similar idea, because they reject outliers by enumerating the groups of three links. However, this method is just a special case of our subgraph enumeration method, because our method can be applied to patch merging as illustrated in Fig. 5.7(b). When there are m shared nodes during patch merging in Fig. 5.4(b), we enumerate $C(m, 3)$ subgraphs that can barely satisfy the globally rigid merging condition. Since there are $n - m$ nonzero-length links, we further enumerate $C(m, 2) \times C(n-m, 1)$ subgraphs that can barely satisfy

the condition in Fig. 5.4(c). Similarly, there are $C(m,1) \times C(n-m,2)$ subgraphs that can barely satisfy the condition in Fig. 5.4(d), and there are $C(n-m,4)$ subgraphs that can barely satisfy the condition in Fig. 5.4(e).

(a) Reject Outlier Link $[B, F]$ (b) Reject Outlier Link $[A, E]$

Fig. 5.8: Reject Outliers by Finding a Minimally Globally Rigid Merging Subgraph without Detectable Outliers.

Examples. We exemplify how we apply the subgraph enumeration method to identify outliers during patch merging.

a. In Fig. 5.8(a), we can identify link $[B, F]$ as an outlier link, when merging patch $\{A, B, C\}$ with patch $\{D, E, F\}$. Because these two patches can be merged without detectable outliers if using the four links $[A, F]$, $[C, F]$, $[B, D]$, $[B, E]$, which satisfies the condition in Fig. 5.4(e).

b. In Fig. 5.8(b), we can identify link $[A, E]$ as an outlier link, when merging generically rigid patch $\{A, B, C, D\}$ with patch $\{C, E, F\}$. Because these two patches can be merged without detectable outliers if using shared node C and two links $[B, F]$, $[D, E]$, which follows the patch merging case in Fig. 5.4(d).

With the above examples, an outlier rejection method can work by two steps: (1) enumerate minimally globally rigid merging subgraphs as shown in Fig. 5.7 and check whether they have detectable outliers; (2) whenever a subgraph $\ddot{G}$ is found without detectable outliers, stop the enumeration and declare all the links that are

consistent with $\ddot{G}$ to be normal links. Link-subgraph consistency relation is defined in Definition 9 that the subgraph with the link added has no detectable outliers.

Definition 9 (Link-Subgraph Consistency)**.** *Given a globally rigid merging subgraph $\ddot{G}$ as in Definition 8, a link ℓ in link set $\mathcal{L}$ is consistent with $\ddot{G}$ which is noted as $\ell \triangleright \ddot{G}$, if*

(1) $\ddot{G}$ has no detectable outliers and

(2) $\ddot{G}$ with link ℓ added has no detectable outliers.

Link ℓ is trivially consistent with $\ddot{G}$, if ℓ is contained in $\ddot{G}$.

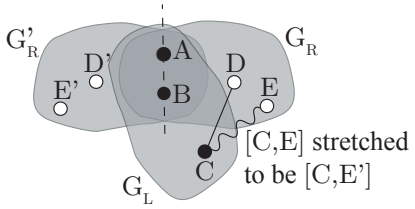
5.3.3 Reliable Outlier Link Rejection against Collusion

However, the above outlier rejection method can fail in two situations where there is collusion.

(1) Existence of an outlier link that colludes with normal links due to geometry effects. In such a situation, we can find a minimally globally rigid subgraph that contains the outlier link and the normal links that collude with it. In this subgraph, we cannot detect the outlier link. For example, in Fig. 5.9(a), link $[C, E]$ is an outlier link declaring its length equal to $[C, E']$. Then we can find a minimally globally rigid subgraph that contains shared nodes A, B and outlier link $[C, E]$. In this subgraph we cannot detect the outlier because there are no abnormal deformations when the right patch is realized to G'_R.

(2) Existence of multiple outliers that collude with each other. We thus may find a minimally globally rigid subgraph with all its links to be colluding outliers, and we cannot detect them. For example, in Fig. 5.9(b), three anchors A, B, C are outliers which declare their positions to be A', B', C' in the same coordinate frame. We cannot

detect the outlier anchors because there are no abnormal deformations when the right patch is realized to G'_R. We delay the discussion to Section 5.5 on how to tolerate the multiple colluding outlier anchors.



(a) Outlier link $[C, E]$ colludes with shared nodes A, B

(b) Colluding outlier anchors A, B, C that declares positions A, B, C' in the same coordinate frame

Fig. 5.9: Colluding Outliers to Justify Voting Algorithm. (a) Outlier link $[C, E]$ are non-rejectable due to flip ambiguity G'_R. (b) Outlier anchors A, B, C are non-detectable.

We present in Algorithm 5.1 the pseudocode of our robust patch merging operation to detect outliers and reject them. This operation, if it cannot reject the outliers due to collusion as shown in Fig. 5.9(a), can report such inability explicitly. The basic idea to beat collusion is by voting that can exploit the majority of normal links, which has four steps: (1) enumerate minimally globally rigid merging subgraphs and check whether they have detectable outliers; (2) when a subgraph has no detectable outliers, calculate the votes earned by this subgraph, and this subgraph can get the vote from a link if this link is consistent with this subgraph, as defined in Definition 9; (3) the subgraph with the highest votes wins, because the subgraph with only normal links will be supported by all normal links, which outnumber outlier links; (4) when multiple subgraphs get the same highest votes, we can identify outliers if these subgraphs have the same set of supporters which is regarded as normal links. Otherwise, we report the inability to identity the outliers since there are multiple subgraphs getting the same highest votes but with different sets of supporters.

For example, Algorithm 5.1 can explicitly report the inability to reject the detected outlier in Fig. 5.9(a) where the outlier link $[C, E]$ colludes with normal links due to its special geometry. We cannot reject outlier link $[C, E]$, because the two realizations G_R and G'_R get the same votes (i.e. three votes) but have different sets of supporters. G_R is supported by shared nodes A, B and normal link $[C, D]$. G'_R is supported by shared nodes A, B and outlier link $[C, E]$.

Table 5.1: Robust Patch Merging based on Voting

Input: Merging topology G is comprised of left patch G_L, right graph G_R (that can be either a sensor or a patch), and a set L of n links between G_L and G_R, as shown in Fig. 5.7.
Output: A larger resultant patch by merging G_L and G_R.

Procedure:
perform the merging of G_L and G_R with connecting links L
if *no detected outliers*, **then return** the merging result directly
initialize max votes $v_{\max}$ as 0 and winner set W as $\emptyset$
foreach subgraph $\ddot{G}$ of input topology G as shown in Fig. 5.7

1. **if** $\ddot{G}$ *is not globally rigid merging*, **then continue**
2. **if** $\ddot{G}$ *is detected to contain outliers*, **then continue**
3. for candidate $\ddot{G}$, calculate its votes v obtained from the set of voters L as $v = \sum_{\ell \in L} \ell \triangleright \ddot{G}$, where $\ell \triangleright \ddot{G}$ means the link ℓ ($\in L$) is consistent with the subgraph $\ddot{G}$
4. **if** $v < v_{\max}$, **then continue**
5. **if** $v == v_{\max}$, **then** $v_{\max} = v$ and W is reset to empty
6. add subgraph $\ddot{G}$ to the winner set W

if W *is empty, or in nonempty* W *the winners have different set of supporters*, **then return** an empty resultant patch;
else regard the shared set of supporters for W as normal links, remove the identified outliers †, use only normal links to merge G_L and G_R, and **return** the merging result.

† Removal of a shared node or a zero-length link is to remove all the edges incident to the shared node.

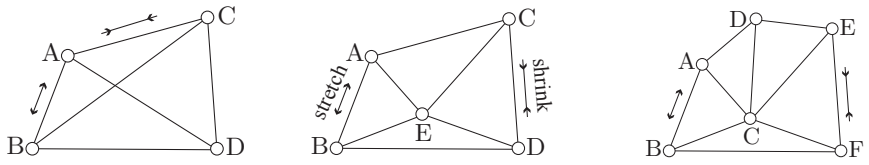
5.4 Robust Network Localization against Non-Rejectable Outlier Links

This section solves the problem of robustly localizing a network in the presence of outlier links. At the present stage, we assume the absence of outlier anchors. Our basic idea of outlier link rejection is to localize a network by iteratively invoking the robust patch merging operation in Algorithm 5.1. However, the challenge is the robust patch merging operation may sometime fail to reject the detected outlier links, either due to collusion or due to insufficient connectivity, which is to be elaborated in Section 5.4.1. This challenge is to be addressed in Section 5.4.2 by recording the failure of rejection and isolating the non-rejectable outlier links.

5.4.1 Non-Rejectable Outlier Links in Patch Merging

The challenge to achieve robust network localization is that sometimes when merging two patches, we can detect outlier links but cannot reject them. This usually happens in sparse subregions of a network containing outlier links. Such sparse subregions are called minimally globally rigid patches, as illustrated in Fig. 5.10 and as defined in Theorem 8. If we do not properly handle the non-rejectable outlier links in such sparse subregions, the final location estimates can be biased.

Theorem 8 (Minimally Globally Rigid Patch & Outlier Link Non-Rejectability)**.** *A patch G is minimally globally rigid, if G is globally rigid and G becomes no longer globally rigid upon the removal of any of its edge. If an outlier link is contained in a minimally globally rigid patch G, then this outlier link can be detected but cannot be rejected.*



(a) Quadrilateral (b) Five-Node Wheel (c) Six-Node Wheel

Fig. 5.10: Non-rejectable outlier links in minimally globally rigid patches. In such patches, deformation of a link incurs deformation of another link. Thus we cannot tell which link is the outlier.

Proof. Intuitively, if patch G is minimally globally rigid, then G has the necessary redundancy in connectivity to detect outliers, but G is not redundant enough to identify them. This is because G becomes generically rigid if one of its edge is removed; for a generically rigid graph, if one of its edge is removed, some part of it is flexible. For example, if links $[A, B]$ and $[A, C]$ are removed in Fig. 5.10(a), node A can rotate about node D. Thus for these two links, the stretch of one leads to the shrink of the other. We can detect the outlier by abnormal deformations but cannot tell which one is an outlier. □

According to Theorem 8, when an outlier link exists during the merging of two patches G_L and G_R, this outlier link cannot be rejected if the resultant patch is minimally globally rigid. Such two patches G_L and G_R are incompatible, according to Definition 10. There exists another situation that two patches are incompatible with a non-rejectable outlier link, i.e. the collusion of the outlier link with normal links as shown in Fig. 5.9(a).

Definition 10 (Patch Incompatibility Relation)**.** *Two patches G_L and G_R are defined to be incompatible, if their merged patch contains an outlier link that cannot be rejected.*

We explain why we have to merge the depicted patch pairs in Fig. 5.8 to reject outlier links. This is because other patch pairs that can be merged globally rigidly are incompatible. In Fig. 5.8(a), patch $\{A, B, C\}$ is incompatible with patch $\{A, C, F\}$, because their resultant patch $\{A, B, C, F\}$ is minimally globally rigid and contains the outlier link; patch $\{B, D, F\}$ is incompatible with patch $\{D, E, F\}$ for the same reason. In Fig. 5.8(b), patch $\{A, B, C\}$ is incompatible with patch $\{C, E, F\}$, because the resultant patch $\{A, B, C, E, F\}$ is minimally globally rigid and contains the outlier link; patch $\{A, C, D\}$ is incompatible with patch $\{A, D, E\}$ for the same reason.

Our robust patch merging operation in Algorithm 5.1 can report the patch incompatibility relation (by returning an empty resultant patch), for the following two reasons. (1) Algorithm 5.1 can discover patch incompatibility due to the lack of redundancy in connectivity shown in Fig. 5.10. Firstly, Algorithm 5.1 can detect the outlier link because the merged patch is minimally globally rigid and the merging of G_L,G_R is globally rigid merging. Secondly, Algorithm 5.1 cannot identify the outlier link because we cannot find an outlier-free subgraph that is globally rigid. Algorithm 5.1 thus reports such non-rejectability by returning an empty merged patch. (2) Algorithm 5.1 can discover patch incompatibility due to the collusion between outlier link with normal links as shown in Fig. 5.9(a), which is already explained in Section 5.3.3.

5.4.2 Robust Network Localization

We present in Algorithm 5.2 our network localization algorithm based on iterative patch merging. This algorithm is robust with the presence with outlier links, because

it

- removes the rejectable outlier links by robust patch merging operation (i.e. Algorithm 5.1) and

- isolates the non-rejectable outlier links when robust patch merging operation reports patch incompatibility relation.

Algorithm 5.2 is composed of the following two phases:

Phase 1 divides the network into basic patches, including a global patch and many local patches. Each local patch can be either a triangle or an individual node.

Phase 2 merges these patches iteratively and each local patch that is merged into the global patch can be localized finally. All cases for two patches to be merged are already shown in Fig. 5.4 for globally rigid merging and in Fig. 5.5 for generically rigid merging. At step 1 of phase 2, we merge two patches G_L and G_R by our robust patch merging operation in Algorithm 5.1. This operation can reject outlier links. Several simulations of outlier link rejection can be found in Section 5.6.3.

Algorithm 5.2 can isolate non-rejectable outlier links, because when two patches are incompatible with non-rejectable outliers, robust patch merging operation in Algorithm 5.1 can detect such a situation. Then at step 2 of phase 2, we memorize the detected incompatible patch pair, and afterwards we will never attempt to merge this incompatible patch pair thanks to the guarding condition of `foreach` loop of phase 2. Finally, Algorithm 5.2 can guarantee the globally rigid subgraphs of the global patch is free from detectable outlier links. Only in these subgraphs, the nodes can be localized and they are immune to the adverse impact of outlier links.

5.5 Robust Network Localization against Outlier Anchors

This section solves the problem of robustly localizing a network in the presence of outlier anchors. Firstly, Section 5.5.1 exposes an inadequacy of the robust network localization algorithm in Algorithm 5.2 that it cannot reliably reject outlier anchors, even when there is only one outlier anchor in the network. Then Section 5.5.2 presents an enhancement to Algorithm 5.2 to reliably reject multiple outlier anchors which may collude due to malicious attacks. Finally, we analyze the robustness of the proposed RobustLoc algorithm against various attacks.

5.5.1 Rejection of Single Outlier Anchor

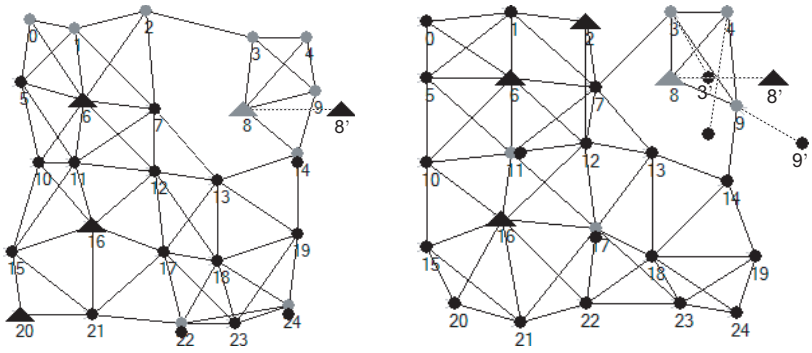
The network localization algorithm presented in Algorithm 5.2 in fact cannot reliably reject a single outlier anchor in a network. This inadequacy is not obvious because it sounds reasonable that an outlier anchor is equivalent to a normal anchor whose incident links are all outlier links. Then a robust network localization algorithm, if it can reject links with abnormal deformations, can remove the normal links incident to outlier anchors and thus automatically reject outlier anchors.

However, we point out that the removal of normal links incident to outlier anchors may cause two problems, even when there is only one outlier anchor in a network. We illustrate these two problems in Fig. 5.11 and describe them below.

1) Reduced Localization Percentage: In Fig. 5.11(a), anchor 8 is an outlier. If normal links $[7, 8], [13, 8]$ incident to the outlier anchor 8 are removed, patch $\{3, 4, 8, 9\}$ cannot be uniquely localized, because it is connected to the network by just three

links $[4, 8]$, $[4, 9]$ and $[2, 3]$.

2) Localized to Wrong Realizations: In Fig. 5.11(b), with the removal of normal links $[7, 8], [13, 8]$, patch $\{3, 4, 8, 9\}$ is localized to the depicted wrong realization $\{3', 4', 8', 9'\}$. Note that this wrong realization has no detectable outliers since it has no abnormally deformed links, i.e. distance $[7, 3]$ is equal to $[7, 3']$, and distance $[14, 9]$ is equal to $[14, 9']$.



(a) Problem 1: Patch $\{3, 4, 8, 9\}$ cannot be localized.

(b) Problem 2: Patch $\{3, 4, 8, 9\}$ is localized to a wrong realization.

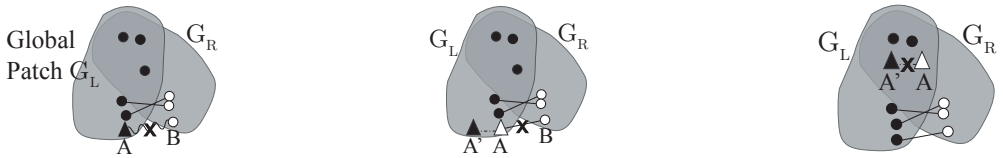
Fig. 5.11: Two Problems of Incorrectly Removing Normal Links $[7, 8]$ and $[13, 8]$ Incident to Outlier Anchor 8.

Considering the above two harms of rejecting normal links incident to outlier anchors, we need to precisely rejecting outlier anchors and we describe the condition as follows.

Theorem 9 (Precise Rejection of One Outlier Anchor during Patch Merging)**.** *During the merging of global patch G_L with a local patch G_R, the condition for the precise rejection of an outlier anchor without the removal of its incident normal links is that the outlier anchor is a shared node between G_L and G_R, like the anchor A shown in Fig. 5.12(c).*

Proof. When merging global patch G_L with a local patch G_R, if the outlier anchor is only incident to a link connecting the two patches as in Fig. 5.12(b), then it is impossible to tell whether this abnormal deformation of link $[A, B]$ is caused an outlier anchor A as in Fig. 5.12(b) or is caused by an outlier link $[A, B]$ as in Fig. 5.12(a). If we simply regard link $[A, B]$ as an outlier link, then the removal of normal links may cause two problems, as described before.

In contrast, if the outlier anchor is a shared node between two patches in Fig. 5.12(c), we can know unambiguously that anchor declaration A' is erroneous, because A' is contained in global patch G_L, node A is contained in local patch G_R, and zero-length link $[A', A]$ is abnormally stretched. □



(a) Reject outlier link $[A, B]$ (b) Falsely reject normal link $[A, B]$ (c) Reject outlier anchor A

Fig. 5.12: Reliable Rejection of Single Outlier Anchor. In both (a) and (b), abnormal deformation can be found in link $[A, B]$. But we cannot tell whether this deformation is caused by outlier link $[A, B]$ as shown in (a), or by outlier anchor A as shown in (b).

5.5.2 Rejection of Multiple Outlier Anchors

The rejection of multiple colluding outlier anchors is different from the rejection of a single outlier anchor, because *local knowledge* may not be enough to reject the colluding outlier anchors. Here, "local knowledge" means that not all anchors are involved in the merging of the global patch and a local patch. For example, in Fig. 5.13(a), we can reject the outlier anchor A, when we merge global patch G_0 with

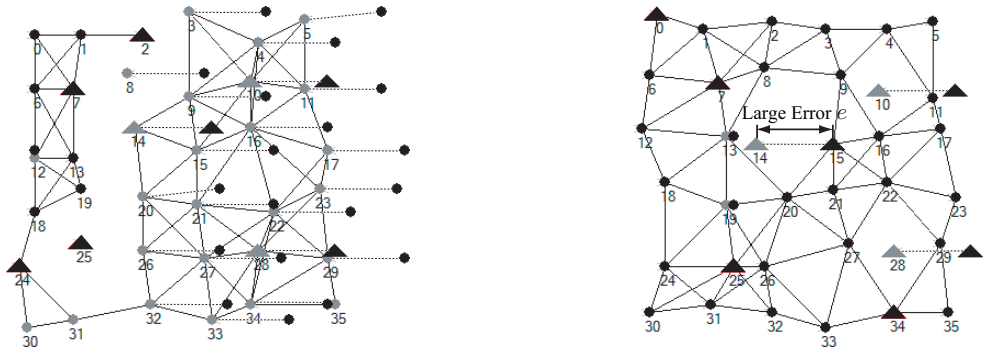
local patch G_1, under the condition that anchor A is a shared node between G_0 and G_1 (note that all depicted links in Fig. 5.13 can be either nonzero-length links or zero-length links). However, global patch G_0 contains three colluding outlier anchors A, B, C which declare their positions to A', B', C' in the same coordinate frame. The local patch G_2, when it is merged into global patch G_0, can be deceived by the three outlier anchors to be localized in their coordinate frame, since G_2 has most of its links connecting to them.

(a) Patches G_1, G_2 with Local Knowledge (b) Patch G_3 with Global Knowledge

Fig. 5.13: Networks with multiple outlier anchors. Note that a link depicted in this figure represents a shared node if the link length is zero.

We verify by simulations that small local patches only with local knowledge can be deceived by colluding outlier anchors. In Fig. 5.14(a) with three colluding outlier anchors $10, 14, 28$, all the sensors in the right part of the network are shifted rightwards to be localized in the coordinate frame of these outlier anchors. In contrast, in the left part of the network where the normal anchors dominate, sensors are still localized in the correct coordinate frame.

When there exist multiple colluding outlier anchors, we need network-wide *global knowledge* to reliably reject them. Here, "global knowledge" (as formally defined in Definition 11) means that the local patch G_3 shown in Fig. 5.13(b) has links to all anchors during the merging with global patch G_0. Patch G_3 thus has enough information to exploit the network-wide majority of normal anchors over outlier anchors. This



(a) Nodes are cheated by colluding outlier anchors. (b) Successful rejection of colluding outlier anchors.

Fig. 5.14: Localization of networks with four normal anchors and three colluding outlier anchors 10, 14, 28. (a) is not configured with delayed global patch merging feature. (b) is configured with such a feature.

majority voting algorithm is already adopted by our robust patch merging algorithm in Algorithm 5.1.

Definition 11 (Global Knowledge for Reliable Rejection of Outlier Anchors)**.** *Multiple colluding outlier anchors can be rejected reliably by our robust patch merging operation, under the following conditions. (1) Given link set L connecting the global patch G_L and a local patch G_R, for each anchor in the global patch G_L, there is a link in L incident to it as shown in Fig. 5.13(b). (2) In the link set L, the links incident to normal anchors are the majority.*

To reliably reject colluding outlier anchors, we propose an improvement (named *delayed global patch merging*) to the iterative patch merging process in Algorithm 5.2. The basic idea is to construct a large local patch having the global knowledge that satisfies the condition in Definition 11. When merging this large local patch (with global knowledge) into the global patch, we can differentiate normal anchors from outlier anchors by the voting algorithm supported by robust patch merging operation in Algorithm 5.1.

In order to construct this large local patch, we firstly merge local patches iteratively excluding the global patch. When this iterative local patch merging process terminates, we find the largest local patch that remains, which is generically rigid. Then we merge this local patch with global patch using our robust patch merging algorithm in Algorithm 5.1. During this merging, the anchors are shared nodes due to the large size of the local patch. This merging thus can reject the outlier anchors without ambiguities according to Theorem 9. We name such a localization process as *RobustLoc*.

5.5.3 Security Analysis of RobustLoc Algorithm

We analyze the security of our robust network localization algorithm (named *RobustLoc*) against various attack models. This RobustLoc algorithm is Algorithm 5.2 with the improvement of delayed global patch merging in Section 5.5.2.

Outlier Links. Outlier links exist either due to environmental interferences (e.g. non-line-of-sight) or due to launched malicious interferences to distance measurement signals. If these distance outliers exist within the links connecting two patches, our robust patch merging algorithm in Algorithm 5.1 can reject these outliers by finding a trustworthy link subset excluding these outlier links, as shown in Fig. 5.8. If these outlier links are non-rejectable due to the lack of redundancy in network connectivity, Algorithm 5.2 can isolate them.

Single Compromised Anchor. We have a compromised anchor, if an outside attacker has cracked the cryptographic key and password of this anchor, or if this compromised anchor is in fact a betrayer. We assume that each anchor has its unique cryptographic key, otherwise a single betrayer knowing the shared key can wreck the

whole network. We can tolerate the compromised anchor, because we can reliably reject this single outlier anchor during patch merging when this anchor is a shared node as shown in Fig. 5.12(c).

Replay Attack or Sybil Attack. An attacker can launch the Replay attack, if it overhears an anchor declaration and replays this declaration at other places. An attacker can launch the Sybil attack, if it grabs a compromised anchor, exploits this anchor's identity and makes falsified anchor declarations at different places [Newsome et al., 2004]. Algorithm 5.2 can defeat both Replay attacks and Sybil attacks, because its input anchor set A_N implicitly assumes that each anchor identity corresponds to only one position declaration. The duplicated position declarations with the same anchor identity can be compressed to one declaration and get tolerated by Algorithm 5.2. A safer solution perhaps is for the base station to revoke this compromised identity upon the detection of a cloned anchor identity. Sybil attack with multiple forged identities can be treated as the multiple compromised anchors attack as follows.

Multiple Compromised Anchors. This attack is the most troublesome since the attacker may adopt the strategy that firstly pursues local superiority and finally achieve whole-field superiority, i.e. although initially the multiple comprised anchors cannot outnumber benign anchors in the whole field, the comprised anchors can outnumber benign anchors in a particular small region. The sensors in that small region can get deceived, since they have most of their distance measurements incident to the compromised anchors, which is already shown in Fig. 5.14(a). Our simulation in Fig. 5.14(b) shows that our RobustLoc can reliably reject such multiple compromised anchors with local superiority. This is because RobustLoc algorithm uses the global

knowledge to beat multiple compromised anchors as shown in Fig. 5.13(b).

5.6 Simulation Results

This section verifies the robustness of our RobustLoc algorithm against outlier links and outlier anchors, in various network conditions. Section 5.6.1 briefly describes simulation settings. Section 5.6.2 verifies that our robust patch merging operation in Algorithm 5.1 can effectively detect and reject outlier links. Section 5.6.3 shows that our robust network localization algorithm *RobustLoc* can achieve higher localization percentage than RobustMultilateration [Kiyavash and Koushanfar, 2007] in sparse networks. Section 5.6.4 shows that our RobustLoc algorithm is more robust and accurate than state-of-the-art localization algorithm CALL [Wang et al., 2008] by rejecting outlier links and outlier anchors. Finally in Section 5.6.5, we show that RobustLoc can provide good localization accuracy and high localization percentage in concave networks.

5.6.1 Simulation Settings

Our simulations assume the following system parameters. We assume Cricket for the ranging system, which is based on ultrasonic TOA. Thus the ranging noise σ is configured as 2 cm and the ranging radius r is set to 6 m, as listed in the following table. For the ranging noise model, we adopt the empirical model in [Whitehouse et al., 2005] with heavy tails to overestimate distance probably due to non-line-of-sight conditions. For outlier distances and outlier anchors with abnormally large error, their error magnitude e can be adjusted. For network topology, nodes are

arranged by a disturbed grid. By adjusting grid spacing s, we can generate networks with different degree d.

Configured Parameters of Wireless Network Localization	
σ	expected noise of ranging methods, configured as 0.02m
r	radius to effectively obtain ranging data, configured as 6m
s	spacing of disturbed grid for node deployments
d	network degree (networks with $d<7$ are sparse networks)
e	error magnitude of outlier anchors and outlier links
Compared Variables of Wireless Network Localization	
a	average localization accuracy (i.e. distance from true location to estimated location) of uniquely localizable nodes
p	percentage of uniquely localizable nodes among all nodes

We compare localization accuracy a and localization percentage p of the three localization algorithms below.

1) **RobustLoc**: our robust localization algorithm in Algorithm 5.2 that invokes the robust patch merging operation in Algorithm 5.1 and adopts the delayed global patch merging feature described in Section 5.5.2.

2) **RobustMultilateration**: iterative multilateration algorithm [Savvides et al., 2003b] that invokes robust multilateration in [Kiyavash and Koushanfar, 2007].

3) **CALL** [Wang et al., 2008]: a localization algorithm that iteratively invokes the patch merging operation in Fig. 5.4 & 5.5, without any enhancements of outlier rejection.

5.6.2 Robust Patch Merging against Outliers

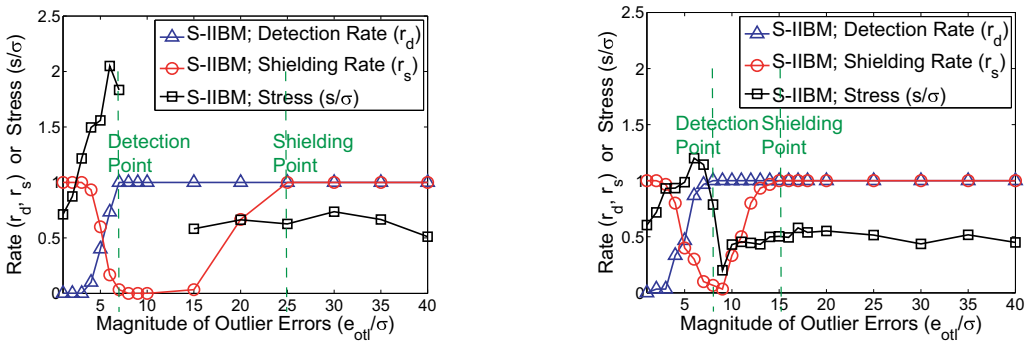
Our first simulation is to investigate the effectiveness of our robust patch merging operation (i.e. Algorithm 5.1) to detect and reject outliers, because Algorithm 5.1 is the primitive of our RobustLoc algorithm. This simulation shows Algorithm 5.1 can detect outliers with nearly 100% success rate ($P_d \approx 1$) when the outlier error e is larger than 7σ, and it can effectively reject outliers and keep the average residue rsd of remaining links below 2σ.

Compared Variable for Patch Merging	
rsd_l	residue of link l, i.e. difference between measured length (e.g. by TOA) and estimated length after patch merging
P_d	the probability of detecting the outliers, i.e. finding any link l have its rsd_l exceeding the threshold $2c\sigma$
rsd	average residue of remaining links with outliers rejected in link set L connecting patches G_L and G_R

Fig. 5.15(a) shows that our robust patch merging operation can effectively detect and reject outliers for the multilateration topology depicted to Fig. 5.7(a). The simulated topology has four normal links and one outlier link. Algorithm 5.1 can detect the existence of the outlier by checking whether the residue rsd_l of any link exceed the threshold $2c\sigma$ where c is a constant. In Fig. 5.15(a), Algorithm 5.1 can detect outliers with nearly 100% success rate ($P_d \approx 1$) when the outlier error e is larger than 7σ. Although the outlier cannot be effectively detected when e is smaller than 7σ, the average residue is still kept below 2σ. When e is larger than 15σ, Algorithm 5.1 can identify the outlier and reject it. Thus the average residue is around 0.5σ. When e is between 7σ and 15σ, Algorithm 5.1 reports that the outlier can be detected but cannot be identified. This is because Algorithm 5.1 can find

multiple winners in the winner set W having the same set supporters.

Fig. 5.15(b) shows that our robust patch merging operation can effectively detect and reject outliers for the patch merging topology as illustrated in Fig. 5.7(b). Outlier rejection in such a topology cannot be handled by RobustMultilateration. The simulated topology has five normal links and one outlier link. In Fig. 5.15(b), the outlier detection rate P_d is nearly 100% when outlier error e is larger than 7σ. The average residue is kept below 1.5σ by rejecting the outlier.



(a) Multilateration Topology in Fig. 5.7(a)

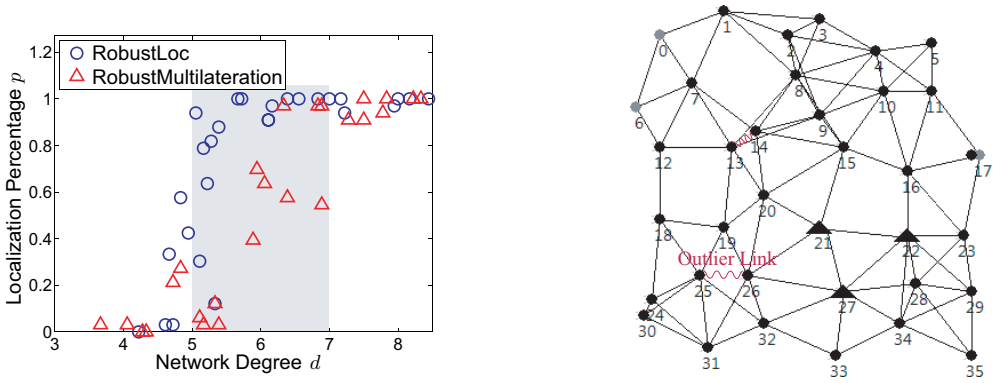
(b) Patch Merging Topology in Fig. 5.7(b)

Fig. 5.15: Robust Patch Merging Operation vs. Outlier Error. (a) shows the rejection in Multilateration topology with one outlier link. (b) shows the rejection in Patch Merging topology with one outlier link.

5.6.3 Outlier Links Toleration in Network Localization

This simulation shows that RobustLoc can effectively tolerate outlier links while achieve high localization percentage in sparse networks. Thus we compare the localization percentage p of RobustLoc and RobustMultilateration. This simulation assumes the presence of outlier links and the absence of outlier anchors in the simulated networks. We also assume that three anchors in the simulated network are

geographically close (like anchors $21, 22, 27$ in Fig. 5.16(b)), since RobustMultilateration needs these nearby anchors to bootstrap the algorithm.



(a) Comparing localization percentage.

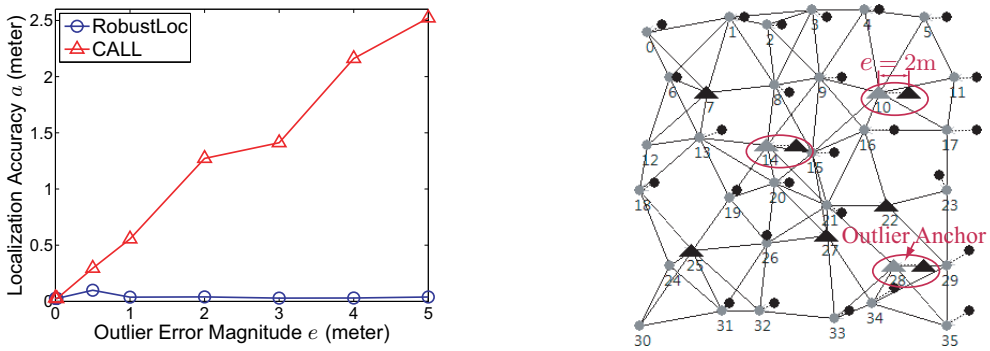
(b) Localization percentage 93%.

Fig. 5.16: Comparison of RobustLoc and RobustMultilateration in sparse networks. (a) A comparison of localization percentage. (b) Localization result of RobustLoc in a sparse network with 5.05 degree and with outlier links $[13, 14], [25, 26]$. In this network, RobustMultilateration cannot bootstrap.

Fig. 5.16(a) shows that, in sparse networks with degree between 5 and 7, localization percentage of RobustLoc is higher than that of RobustMultilateration by at least 30%, because localization percentage of RobustLoc is generally above 80% and that of RobustMultilateration is below 60%. Fig. 5.16(b) depicts a network with 5.05 degree and with two outlier links $[13, 14]$, $[25, 26]$. In such a sparse network, RobustMultilateration cannot localize any nodes because there does not exist a node with three links connecting to anchors $21, 22, 27$. However, if using RobustLoc, localization percentage is as good as 93% and meanwhile the location estimates (i.e. black dots) are close to their true positions.

5.6.4 Toleration of Colluding Outlier Anchors in Network Localization

This simulation shows that RobustLoc can effectively tolerate colluding outlier anchors in sparse networks. We vary the outlier error magnitude e and compare localization accuracy a of RobustLoc and CALL. The simulated networks are sparse networks with degree d around 5.5. In these networks, we deploy seven anchors randomly and three of them are colluding outlier anchors, e.g. $10, 14, 28$ in Fig. 5.17(b).



(a) Impact of outlier error on accuracy.

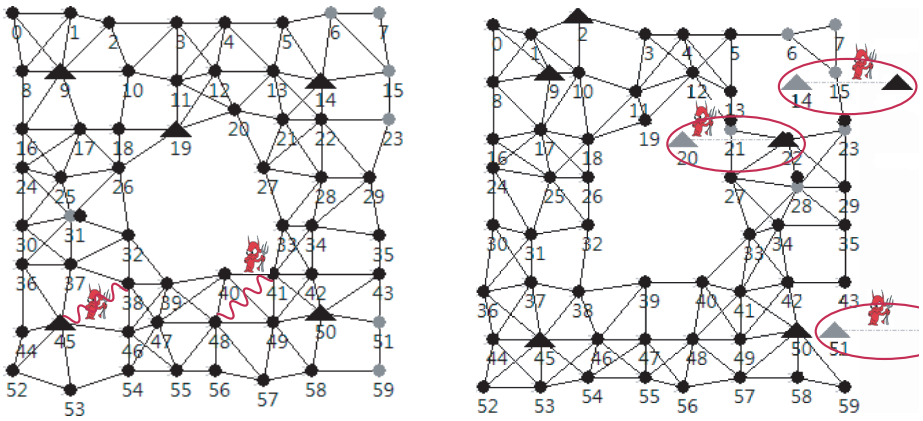
(b) CALL with error $e = 2\,\mathrm{m}$.

Fig. 5.17: RobustLoc vs. CALL in Localization Accuracy a.

Fig. 5.17(a) shows that RobustLoc is robust against outlier anchors but CALL is not. This is because localization accuracy a of RobustLoc is proportional to ranging noise σ regardless of error e of outlier anchors, but accuracy of CALL is proportional to error e. RobustLoc is robust against outlier anchors because RobustLoc can identify and isolate the three outlier anchors as shown in Fig. 5.14(b). CALL is not robust because CALL has no outlier rejection ability. Thus all the sensor nodes shown in Fig. 5.17(b) are dragged from their true positions by the outlier anchors. The magnitude of such deviations are proportional to error e of outlier anchors.

5.6.5 Concave Network Deployment Regions

This simulation verifies that RobustLoc can reliably reject outliers in concave networks. We simulate O-shaped networks as shown in Fig. 5.18, which are sparse networks with degree around 5. Although Fig. 5.18(a) contains two outlier links $[41, 48]$ and $[38, 45]$, our RobustLoc algorithm can still provide good localization accuracy with average error below 2σ and satisfying localization percentage above 80%. Fig. 5.18(b) depicts an O-shaped network with colluding outlier anchors $20, 14, 51$. Our RobustLoc algorithm can effectively identify and isolate these outliers.



(a) Reject Outlier Links; LocError = $1.35\,\sigma$; LocPercentage=89%.

(b) Reject Outlier Anchors; LocError = $1.46\,\sigma$; LocPercentage=94%.

Fig. 5.18: Outlier Rejection Demo in O-Shaped Networks.

5.7 Conclusion

This chapter focuses on the problem of localizing a wireless sensor network robustly against outlier links and outlier anchors. We proposed a solution named RobustLoc which iteratively invokes our robust patch merging operation. Additionally,

we addressed the two challenges of (1) isolating non-rejectable outlier links by finding incompatible patch pairs and (2) reliably rejecting multiple outlier anchors which may collude. Our simulation results show that our RobustLoc algorithm can retain good localization accuracy with the presence of outliers and meanwhile provide high localization percentage in both sparse networks and dense networks.

Table 5.2: Localization of Network $N = \{A_N, E_N\}$

Input: a set of anchors A_N whose locations are already known; a set of links E_N with measured distances.
Output: Location estimates of uniquely localizable nodes.

Procedure:
Phase 1. Divide the Network into Basic Patches, including

- a global patch containing all the anchors A_N,
- local patches, each of which is a triangle,
- local patches, each of which is a single node.

Phase 2. Merge Patches Iteratively.
foreach *patch pair that are not marked incompatible and can satisfy the generically rigid merging condition* †, **do**

1. Merge two patches G_L and G_R connected by link set L (i.e. invoke the robust patch merging operation in Algorithm 5.1)
   - globally rigidly if G_L and G_R can satisfy the condition in Fig. 5.4, or
   - generically rigidly if satisfying the condition in Fig. 5.5

   , and get the larger resultant patch G.
2. **if** *the above merging fails* ‡, **then** mark patch pair $[G_L, G_R]$ as an incompatible pair and **continue** this foreach loop.
3. **if** *either one of patch pair* $[G_L, G_R]$ *is the global patch*, **then** mark the resultant patch G as a global patch ‡†; **else** mark the resultant patch G as a local patch.
4. Destroy the two patches G_L and G_R.

Only the nodes in the global patch can be localized in global coordinate frame. But the global patch may not be globally rigid. Some of its nodes may have unique positions, and others may have ambiguous positions, which is called partial localizability.

† NOTE: If two patches can satisfy globally rigid merging condition, they can also satisfy generically rigid merging condition, and they are given a higher priority to be merged than the patch pairs that cannot satisfy globally rigid merging condition.
‡ Failure to reject detected outliers is indicated by empty resultant patch G obtained at step 1 of phase 2.
‡† If the resultant patch is merged from a local patch and a global patch, it should contain at least three anchors. A generically rigid structure with at least three anchors is a global patch that can be localized ambiguously.

Chapter 6

Conclusions and Future Directions

6.1 Conclusions of Whole Thesis

The research presented in this thesis addresses the localization problem of wireless sensor networks. We explored two categories of localization algorithms: range-free and range-based. The former is much cheaper than the latter. But the precision of range-free localization (i.e. meters or tens of meters) is much coarser than that of range-based localization (i.e. sub-meters or centimeter depending on the ranging methods).

Firstly, we addressed the problem of improving the accuracy of range-free localization in anisotropic sensor networks. Such networks have various anisotropic factors, including irregular radio propagation, anisotropic terrain condition, non-uniform sensor distribution, and concave network shapes. Our presented algorithm is also suitable for large-scale networks, since it can suppress error accumulation by using the nearby anchors to revise remote anchors.

Secondly, we focused on range-based localization and addressed its two important

issues: (1) the toleration of ranging noises to improve localization accuracy, and (2) the enumeration of flip ambiguities to guarantee localization robustness. Previous methods only addressed the two issues for multilateration. We proposed a unified algorithm (called inflexible body merging) that can solve the two issues for both multilateration and patch merging. Thus our algorithm has better performance than previous methods in sparse networks with degree between 4 and 8. Moreover, we presented a unified condition for multilateration and patch merging, which can achieve higher localization percentage than state-of-the-art work.

Finally, we solved the outlier rejection problem for range-based localization. We uncovered the two inadequacies of previous work in this field: (1) add outlier rejection ability only to multilateration and neglect patch merging; (2) overlook the difference between outlier distances and outlier anchors, and thus cannot reject colluding outlier anchors reliably. We proposed a robust network localization algorithm (called RobustLoc) to reject both outlier distances and colluding outlier anchors, in networks which can be either dense or sparse.

6.2 Future Directions

A promising direction is the extension of our range-based localization algorithm to three-dimensional spaces. The challenge is that the ambiguity enumeration problem is much more complicated in 3D than in 2D. Another possible direction is that we can investigate the tracking and navigation problem of mobile robots assisted by the static sensor nodes. These sensor nodes can form a network to provide information that is multihop away to the robots, to reduce the exposure of the robot army to potential threats or to avoid the traffic jamming in narrow corridors.

Appendices

Appendix A

Proofs for Range-Free Localization

A.A Systematical Error of Amorphous

We analyze the systematical error of Amorphous incurred by the last hop distance, which has been plotted in Fig. 3.4. To eliminate the influence of inaccurate *HopSize* and isolate the impact of last hop distance on distance estimation of Amorphous, *HopSize* is assumed to be fixed and accurately known as r, throughout the following analysis.

The systematic error E_2 of Amorphous to estimate last hop distance is the difference between expected estimate $\overline{\hat{d}_{2j}(i)}$ produced by Eq. (3.2) and the real distance value $d_j(i)$. Our analysis is all about how to establish the functional relation from $d_j(i)$ to E_2.

$$E_2 = \overline{\hat{d}_{2j}(i)} - d_j(i). \tag{A.1}$$

The expected estimate $\overline{\hat{d}_{2j}(i)}$ can be calculated by Eq. (A.2), since there are only

three possible values for the hop count $h_j(t)$ of sensor t, which neighbors sensor i. These three possible values are $h_j(i) - 1$, $h_j(i)$ and $h_j(i) + 1$. Moreover, the expected percentage of neighbors of sensor i to have hop count $h_j(i) - 1$, $h_j(i)$ or $h_j(i) + 1$ is equal to the probability, that the hop count $h_j(t)$ is $h_j(i) - 1$, $h_j(i)$ or $h_j(i) + 1$.

$$\begin{aligned}\overline{\hat{d}_{2j}(i)} \quad = \quad & P\left[h_j(t) < h_j(i)\right] \cdot (h_j(i) - 1) \\ & + P\left[h_j(t) = h_j(i)\right] \cdot \ h_j(i) \\ & + P\left[h_j(t) > h_j(i)\right] \cdot (h_j(i) + 1) \\ & - 0.5 \qquad\qquad , \ where\ t \in N(i). \end{aligned} \tag{A.2}$$

The probability $P\left[h_j(t) < h_j(i)\right]$ required by Eq. $(A.2)$ is equal to the division of the area $\mathcal{A}\left(r,\ (h_j(i) - 1)r,\ d_j(i)\right)$ of the intersected region of two disks, by the area $\pi\, r^2$ of sensor i's entire neighbor. The intersecting two disks are sensor i's one hop disk with radius r and anchor j's $h_j(i) - 1$ hop disk, whose radius is approximated to $(h_j(i) - 1)r$. The area of the intersecting region can be calculated, when the distance $d_j(i)$ between the centers of the two disks is known additionally.

$$P\left[h_j(t) < h_j(i)\right] \quad = \quad \frac{\mathcal{A}\left[r,\ (h_j(i) - 1)\, r,\ d_j(i)\right]}{\pi\, r^2}. \tag{A.3}$$

The probability $P\left[h_j(t) \leq h_j(i)\right]$ is calculated in a similar way with $P\left[h_j(t) < h_j(i)\right]$.

$$P\left[h_j(t) \leq h_j(i)\right] \quad = \quad \frac{\mathcal{A}\left[r,\ h_j(i)\, r,\ d_j(i)\right]}{\pi\, r^2}. \tag{A.4}$$

The other two probabilities $P\left[h_j(t) = h_j(i)\right]$ and $P\left[h_j(t) >\ h_j(i)\right]$ needed by Eq. (A.2)

can be calculated as follows.

$$\begin{aligned} P\left[h_j(t) = h_j(i)\right] &= P\left[h_j(t) \leq h_j(i)\right] - P\left[h_j(t) < h_j(i)\right]. \\ P\left[h_j(t) > h_j(i)\right] &= 1 - P\left[h_j(t) \leq h_j(i)\right]. \end{aligned}$$

The above equations help establish a functional relation from $d_j(i)$ to E_2, with $h_j(i) = \lceil \frac{d_j(i)}{r} \rceil$. This functional relation has been plotted in Fig. 3.4 and indicates the underperformance of Amorphous in one or two hops. This is consistent with the simulation results in Subsection 3.7.2.

A.B Error Characteristics of CrMcs

we present a theoretical analysis on the error characteristics of the proposed CrMcs, showing that CrMcs can reduce distance estimation error to below $0.2r$ (sensor density > 8), if applied to the CR pattern. The accuracy of CrMcs primarily depends on two factors:

1. the estimation accuracy about the intersected area $a_j(i)$ - the accuracy of Eq. (3.3).

2. the estimation accuracy about the radius of $dsk_j(i)$ - the accuracy of Eq. (3.4);

Generally speaking, in dense networks and isotropic networks with accurate $HopSize$ estimates, the first factor dominates, which is directly connected with the last hop distance. However, in sparse networks and anisotropic networks, where the estimation of $HopSize$ is inaccurate, the second factor prevails to have a greater impact. Our key point is that no matter a network is sparse or dense (anisotropic or isotropic), the last

hop distance (the first factor) is always an important factor in the low hop count cases (within 3 or 4 hops empirically), in which the error in $HopSize$ estimation (the second factor) has a reduced influence on the anchor-sensor distance estimation accuracy. This is why we recommend to adopt CrMcs for the CR pattern in our localization scheme.

In this subsection, we focus our analysis to the impact of the first factor (the last hop distance), since our localization scheme (in Fig. 3.2) applies CrMcs only to the low hop count cases. Driven by this intention, we fix the $HopSize$ to r to eliminate the impact of inaccurate $HopSize$ and isolate the impact of the last hop distance, which leads to the following simplified expression of $\hat{d}_{3j}(i)$, with $r(1)$ fixed to r and $r(h_j(i))$ fixed to $h_j(i) \cdot r$.

$$\hat{d}_{3j}(i) \quad = \quad \mathcal{A}^{-1}\,[r,\ h_j(i)\cdot r,\ \pi\, r^2 \cdot \frac{|N_j(i)|}{|N(i)|}]. \tag{A.5}$$

In this equation, the only randomized variable that determines $\hat{d}_{3j}(i)$ is $\frac{|N_j(i)|}{|N(i)|}$, which is essentially a Binomial trial testing the percentage of neighbor owning hop count no larger than $h_j(i)$. Hence, the accuracy of CrMcs can be theoretically estimated as the average absolute deviation of $\hat{d}_{3j}(i)$:

$$Dev[\hat{d}_{3j}(i)] \quad = \quad \sum_{|N_j(i)|\ \in\ [0,\ |N(i)|]} C_{|N(i)|}^{|N_j(i)|} \cdot \widetilde{P}^{|N_j(i)|} \cdot\ (1-\widetilde{P})^{|N(i)\,-\,N_j(i)|} \cdot |\hat{d}_{3j}(i) - d_j(i)|,$$

$$where \quad \widetilde{P} \quad = \quad \frac{\mathcal{A}\,[r,\ h_j(i)\cdot r,\ d_j(i)]}{\pi\, r^2}.$$

We have plotted the relation between the error of CrMcs $Dev[\hat{d}_{3j}(i)]$ and the sensor density $|N(i)|$ in Fig. 3.6.

Bibliography

Abraham, I., Dolev, D., and Malkhi, D. (2004). Lls: a locality aware location service for mobile ad hoc networks. In *Proc. of DIALM-POMC*.

Bachir, A., Dohler, M., Watteyne, T., and Leung, K. (2010). MAC essentials for wireless sensor networks. *Communications Surveys Tutorials, IEEE*, 12(2):222 – 248.

Buettner, M., Yee, E., V., G., Anderson, and Han, R. (2006). X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proc. of ACM SenSys*.

Ceriotti, M., Mottola, L., Picco, G. P., Murphy, A., Guna, S., Corra, M., Pozzi, M., Zonta, D., and Zanon, P. (2009). Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *Proc. of IPSN/SPOTS, Best paper award*.

Chung, W. C. and Ha, D. S. (2003). An accurate ultra wideband ranging for precision asset location. In *Proc. IEEE Conf. Ultra Wideband Syst. and Technologies*, pages 389–393.

Culler, D., Estrin, D., and Srivastava, M. (2004). Overview of sensor networks. In *Special Issue in Sensor Networks, IEEE Computer*, pages 41–49.

Dimitrios, L., Quentin, L., and Andreas, S. (2006). An empirical characterization of radio signal strength variability in 3-D ieee 802.15.4 networks using monopole antennas. In *Proc. of EWSN*, pages 326–341.

Doherty, L., pister, K. S. J., and Ghaoui, L. E. (2001). Convex position estimation in wireless sensor networks. In *Proc. of IEEE INFOCOM*, volume 3, pages 1655–1663.

Dutta, P., Dawson-Haggerty, S., Chen, Y., Liang, C.-J. M., and Terzis, A. (2010). Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *Proc. of ACM SenSys*.

Eren, T., Goldenberg, D. K., Whiteley, W., Yang, Y. R., Morse, A. S., Anderson, B. D. O., and Belhumeur, P. N. (2004). Rigidity, computation, and randomization in network localization. In *Proc. of IEEE INFOCOM*.

Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. (1999). Next century challenges: Scalable coordination in sensor networks. In *Proc. of ACM MobiCom*, pages 263–270.

Foy, W. H. (1976). Position-location solutions by Taylor-series estimation. *IEEE Trans. on AES*, AES-12(2):187–194.

Gao, T., Massey, T., Selavo, L., Crawford, D., rong Chen, B., Lorincz, K., Shnayder, V., and Welsh, M. (2007). The advanced health and disaster aid network: A light-weight wireless medical system for triage. *IEEE Transactions on Biomedical Circuits and Systems*.

Goldenberg, D. K., Bihler, P., Cao, M., Fang, J., Anderson, B. D. O., Morse, A. S., and Yang, Y. R. (2006). Localization in sparse networks using sweeps. In *Proc. of ACM MobiCom*, Los Angeles, CA, USA.

He, T., Huang, C., Blum, B. M., Stankovic, J. A., and Abdelzaher, T. (2003). Range-free localization schemes for large scale sensor networks. In *Proc. of ACM MobiCom*.

Horn, B. K. P., Hilden, H., and Negahdaripour, S. (1988). Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*.

Hu, L. and Evans, D. (2004). Localization for mobile sensor networks. In *Proc. of ACM MobiCom*, pages 45–57.

Huang, C.-F. and Tseng, Y.-C. (2005). The coverage problem in a wireless sensor network. *Mob. Netw. Appl.*, 10:519–528.

Ji, X. and Zha, H. (2004). Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling. In *Proc. of IEEE INFOCOM*.

Jian, L., Yang, Z., and Liu, Y. (2010). Beyond triangle inequality: Sifting noisy and outlier distance measurements for localization. In *Proc. of IEEE INFOCOM*.

Jiang, X., Ly, M. V., Taneja, J., Dutta, P., and Culler, D. (2009). Experiences with a high-fidelity wirelessbuilding energy auditing network. In *Proc. of ACM SenSys*.

Jin, M., Xia, S., Wu, H., and Gu, X. (2011). Scalable and fully distributed localization with mere connectivity. In *Proc. of IEEE Infocom*.

Karp, B. and Kung, H. T. (2000). GPSR: greedy perimeter stateless routing for wireless networks. In *Proc. of ACM MobiCom*, pages 243–254.

Kiyavash, N. and Koushanfar, F. (2007). Anti-collusion position estimation in wireless sensor networks. In *Proc. of IEEE MASS*.

Kleinrock, L. and Silvester., J. (1978). Optimum tranmission radii for packet radio networks or why six is a magic number. In *Natnl. Telecomm. Conf.*, pages 431–435.

Ko, Y.-B. and Vaidya, N. H. (2000). Location-aided routing (LAR) in mobile ad hoc networks. In *Proc. of ACM MobiCom*, pages 307–321.

Kumar, S., Lai, T. H., and Balogh, J. (2004). On k-coverage in a mostly sleeping sensor network. In *Proc. of ACM MobiCom*, pages 144–158, New York, NY, USA.

Kung, H. T., Lin, C.-K., Lin, T.-H., and Vlah, D. (2009). Localization with snap-inducing shaped residuals (SISR): coping with errors in measurement. In *Proc. of ACM MobiCom*.

Lennvall, T., Svensson, S., and Hekland, F. (2008). A comparison of wirelesshart and zigbee for industrial applications. In *IEEE International Workshop on Factory Communication Systems*, pages 85–88.

Levis, P., Brewer, E., Culler, D., Gay, D., Madden, S., Patel, N., Polastre, J., Shenker, S., Szewczyk, R., and Woo, A. (2008). The emergence of a networking primitive in wireless sensor networks. *Commun. ACM*, 51(7):99–106.

Levis, P., Madden, S., Gay, D., Polastre, J., Szewczyk, R., Woo, A., Brewer, E., and Culler, D. (2004). The emergence of networking abstractions and techniques in tinyos. In *Proc. of USENIS NSDI*, pages 1–14.

Li, J., Jannotti, J., De Couto, D. S. J., Karger, D. R., and Morris, R. (2000). A scalable location service for geographic ad hoc routing. In *Proc. of ACM MobiCom*, pages 120–130, Boston, Massachusetts.

Li, M. and Liu, Y. (2007). Rendered path: range-free localization in anisotropic sensor networks with holes. In *Proc. of ACM MobiCom*, pages 51–62.

Li, Z., Trappe, W., Zhang, Y., and Nath, B. (2005). Robust statistical methods for securing wireless localization in sensor networks. In *Proc. of ACM/IEEE IPSN*, page 12.

Lim, H. and C., H. J. (2005). Localization for anisotropic sensor networks. In *Proc. of IEEE INFOCOM*, pages 138–149.

Liu, D., Ning, P., and Du, W. K. (2005). Attack-resistant location estimation in sensor networks. In *Proc. of ACM/IEEE IPSN*.

Lorincz, K. and Welsh, M. (2005). Motetrack: A robust, decentralized approach to rf-based location tracking. In *Proc. of the International Workshop on Location and Context-Awareness (LoCA)*.

Mandal, S., Turicchia, L., and Sarpeshkar, R. (2009). A battery-free tag for wireless monitoring of heart sounds. *International Workshop on Wearable and Implantable Body Sensor Networks*, 0:201–206.

Meertens, L. and Fitzpatrick, S. (2004). The distributed construction of a global coordinate system in a network of static computational nodes from inter-node distances. Technical Report KES.U.04.04, Kestrel Inst.

Meguerdichian, S., Koushanfar, F., Potkonjak, M., and Srivastava, M. (2001). Coverage problems in wireless ad-hoc sensor networks. In *Proc. of IEEE INFOCOM*, volume 3, pages 1380–1387.

Molisch, A. F., Balakrishnan, K., chin Chong, C., Emami, S., Fort, A., Karedal, J., Kunisch, J., Schantz, H., Schuster, U., and Siwiak, K. (2004). Ieee 802.15.4a channel model - final report. In *Converging: Technology, work and learning. Australian Government Printing Service*. [Online Available].

Moore, D., Leonard, J., Rus, D., and Teller, S. (2004). Robust distributed network localization with noisy range measurements. In *Proc. of ACM SenSys*.

Moteiv (2006). Tmote sky datasheet: `http://www.bandwavetech.com/download/tmote-sky-datasheet.pdf`.

Nagpal, R., Shrobe, H., and Bachrach, J. (2003). Organizing a global coordinate system from local information on an ad hoc sensor network. In *Proc. of ACM/IEEE IPSN*.

Newsome, J., Shi, E., Song, D., and Perrig, A. (2004). The sybil attack in sensor networks: analysis & defenses. In *Proc. of ACM/IEEE IPSN*.

Niculescu, D. and Nath, B. (2003). DV based positioning in ad hoc networks. *Kluwer jrnl of Telecommunication Systems*.

Pages-Zamora, A., Vidal, J., and Brooks, D. H. (2002). Closed-form solution for positioning based on angle of arrival measurements. In *Proc. of IEEE PIMRC*, pages 1522–1526.

Pister, K. and Doherty, L. (2008). TSMP: Time synchronized mesh protocol. In *International Symposium on Distributed Sensor Networks (DSN)*.

Polastre, J., Hill, J., and Culler, D. (2004). B-MAC: Versatile low power media access for wireless sensor networks. In *Proc. of ACM SenSys*.

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

Priyantha, N. B., Balakrishnan, H., Demaine, E., and Teller, S. (2003). Anchor-free distributed localization in sensor networks. In *Proc. of ACM SenSys*.

Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. (2000). The Cricket location-support system. In *Proc. of ACM MobiCom*, pages 32–43.

Priyantha, N. B., Miu, A. K., Balakrishnan, H., and Teller, S. (2001). The cricket compass for context-aware mobile applications. In *Proc. of ACM MobiCom*, pages 1–14.

Rhee, I., Warrier, A., Aia, M., and Min, J. (2005). Z-MAC: a hybrid MAC for wireless sensor networks. In *Proc. of ACM SenSys*.

Savvides, A., Garber, W. L., Adlakha, S., Moses, R. L., and Srivastava, M. B. (2003a). On the error characteristics of multihop node localization in ad-hoc sensor networks. In *Proc. of ACM/IEEE IPSN*.

Savvides, A., Han, C.-C., and Strivastava, M. B. (2001). Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proc. of ACM MobiCom*, pages 166–179.

Savvides, A., Park, H., and Srivastava, M. B. (2003b). The n-hop multilateration primitive for node localization problems. *MONET*, pages 443–451.

Shang, Y. and Ruml, W. (2004). Improved MDS-based localization. In *Proc. of IEEE INFOCOM*, volume 4, pages 2640–2651.

Shang, Y., Ruml, W., Zhang, Y., and Fromherz, M. P. J. (2003). Localization from mere connectivity. In *Proc. of ACM MobiHoc*.

Smith, A., Balakrishnan, H., Goraczko, M., and Priyantha, N. (2004). Tracking moving devices with the cricket location system. In *In Proc. of USENIX/ACM MobiSys*, pages 190–202.

Song, J., Han, S., Mok, A., Chen, D., Lucas, M., Nixon, M., and Pratt, W. (2008). Wirelesshart: Applying wireless technology in real-time industrial process control. *Real-Time and Embedded Technology and Applications Symposium, IEEE*, 0:377–386.

Stoleru, R., He, T., and Stankovic, J. A. (2007). Range-free localization. In *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, volume 30, pages 3–31. Springer US.

Sun, Y., Gurewitz, O., and Johnson, D. B. (2008). RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proc. of ACM SenSys*.

Sundararaman, B., Buy, U., and Kshemkalyani, A. D. (2005). Clock synchronization for wireless sensor networks: A survey. *Ad Hoc Networks (Elsevier)*, 3:281–323.

Szewczyk, R., Osterweil, E., Polastre, J., Hamilton, M., Mainwaring, A., and Estrin, D. (2004). Habitat monitoring with sensor networks. *Communications of the ACM*, 47:34–40.

Volgyesi, P., Balogh, G., Nadas, A., Nash, C. B., Ledeczi, A., Pence, K., Bapty, T., Scott, J., and Police, T. N. (2007). Shooter localization and weapon classification with soldier-wearable networked sensors. In *Conference on Mobile Systems, Applications, and Services (MobiSys)*.

Wang, C., Liu, A., and Ning, P. (2007). Cluster-based minimum mean square estimation for secure and resilient localization in wireless sensor networks. In *Proc. of IEEE WASA*, pages 29–37.

Wang, C. and Xiao, L. (2006). Locating sensors in concave areas. In *Proc. of IEEE INFOCOM*, pages 1–12.

Wang, X., Luo, J., Li, S., Dong, D., and Cheng, W. (2008). Component based localization in sparse wireless ad hoc and sensor networks. In *Proc. of IEEE ICNP*.

Werner-Allen, G., Lorincz, K., J.Johnson, Lees, J., and Welsh., M. (2006). Fidelity and yield in a volcano monitoring sensor network. In *Proc. of OSDI*.

Whitehouse, K., Karlof, C., Woo, A., Jiang, F., and Culler, D. (2005). The effects of ranging noise on multihop localization: an empirical study. In *Proc. of ACM/IEEE IPSN*.

Witkin, A., Baraff, D., and Kass, M. (1997). Physically based modeling: Principles and practice. In *ACM SIGGRAPH Course Notes*, pages 1–12.

Xiao, B., Chen, H., and Zhou, S. (2008). Distributed localization using a moving beacon in wireless sensor networks. *IEEE Trans. on Parallel and Distributed Systems*, 19:587–600.

Xiao, B., Chen, L., Xiao, Q., and Li, M. (2010a). Reliable anchor-based sensor localization in irregular areas. *IEEE Trans. on Mobile Computing*, 9:60–72.

Xiao, Q., Xiao, B., Cao, J., and Wang, J. (2010b). Multihop range-free localization in anisotropic wireless sensor networks: A pattern-driven scheme. *IEEE Trans. on Mobile Computing*, 9:1592–1607.

Ye, W., Heidemann, J., and Estrin, D. (2002). S-MAC: An energy-efficient MAC protocol for wireless sensor networks. In *Proc. of IEEE Infocom*, pages 1567–1576.

Ye, W., Silva, F., and Heidemann, J. (2006). Ultra-low duty cycle MAC with scheduled channel polling. In *Proc. of ACM SenSys*, pages 321–334.

Zhang, Z., Zhou, X., Zhang, W., Zhang, Y., Wang, G., Zhao, B. Y., and Zheng, H. (2011). I am the antenna: Accurate outdoor ap location using smartphones. In *Proc. of ACM MobiCom*.