OBSERVATION PROBLEMS INVOLVING WIRELESS SENSOR NETWORKS (DRAFT 7)

by

Zhen Song

A dissertation proposal submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Electrical Engineering

Approved:

Dr. YangQuan Chen Major Professor	Dr. Brandon Eames Committee Member
Dr. Jake Gunther Committee Member	Dr. Wei Ren Committee Member
Dr. Vlademir Kulyukin Committee Member	

UTAH STATE UNIVERSITY
Logan, Utah

2007

To my family, for their encouragements and supports.

Abstract

Observation Problems Involving Wireless Sensor Networks (draft 7)

by

Zhen Song, Doctor of Philosophy

Utah State University, 2007

Major Professor: Dr. YangQuan Chen
Department: Electrical and Computer Engineering

An important application of wireless sensor networks is to observe the physical world. This dissertation focuses on observation methods by wireless sensor networks (WSNs), especially the unique issues due to the properties of WSNs.

Mobility is an important feature of WSNs. Since the quality of the observation on the physical world depends on the positions of the sensors, the optimum sensor positions depend on the models of the physical systems. Mobile sensor networks can provide better estimates, since the sensor nodes can maneuver to the optimal sensing positions. In Chapter 2, we study a problem called trajectory optimization for observation on Distributed Parameter Systems (DPSs). Given a model of a distributed parameter system, we proposed a numeric method to compute the optimal trajectory of mobile sensors, which are mounted on differentially-driven robots, such that the observation on the system parameter is optimized.

Energy efficiency is the key challenge and the central research topic for WSNs. For WSN-based observation system, it is important to balance the tradeoff between the energy efficiency and estimation precision. Energy costs can be reduced significantly by selecting proper sensors. In Chapter 3, we propose a convex optimal sensor selection (COSS) algorithm to choose proper sensors to observe parameters of physical systems, such as the

position of a target. After introducing several approximations, we formulate the sensor selection problem as a convex optimization problem. Unlike common optimization formulations, we do not explicitly minimize the number of selected sensors. Our cost function is constructed to minimize estimation errors. However, due to the properties of the convex optimization, the number of selected sensors is minimized simultaneously during the time when the estimation error is minimized, which is referred as the implicit optimal sensor selection (IOSS). Our analysis provides some guidance on how to design and tune the IOSS methods. We prove that those IOSS methods not only select the minimum number of sensors that allowed by the Carathéodory's theorem, but also approximate the minimal estimation error predicted by the Cramér-Rao lower bound (CRLB). In addition, we have tested the COSS algorithm by extensive simulation and hardware experiments. Those experiments are conducted under realistic conditions that are close to the real-world engineering practices.

Localization is a fundamental problem for WSNs. The sensors' positions are required by most of the high-level WSN applications. In Chapter 4, an asynchronous TDOA localization method is proposed. When there are a large number of sensor nodes whose positions are unknown, asynchronous TDOA methods reduce communication costs as well as the beacon nodes' hardware costs. The positioning error of the localization system is affected by the placement of the beacon notes. In order to guarantee the robustness of the localization system, the beacon placement problem is formulated as a min-max optimization. The optimization is solved by semi-infinite programming method.

Although the above problems are not identical, they share the same theoretical framework, i.e, optimal experimental design. They are formulated as convex optimization on cost functions based on the Fisher information matrix (FIM). FIM and convex analysis are promising mathematical tools for WSN-based signal processing.

(169 pages)

Acknowledgments

This dissertation is a summary of my research in Utah State University and Siemens Corporate Research. In Utah State University, my research was under the guidance of my PhD advisor, Dr. YangQuan Chan, and my Master advisor, Dr. Kevin Moore. During my internship in Siemens Corporate Research, Dr. Chellury Ram Sastry gave me many insightful advices. They always encourage me to focus on the fundamental problems and continuously improve my ideas by critical thinking. This is the best education experiences I have ever had in my life. I sincerely appreciate their time and efforts on helping me, including the PhD advices as well as the training on my mind.

I feel really lucky since there are so many people who helped me on my research. Dr. Dariusz Uciński, Dr. Ren Wei, Dr. Vladimeir Kulyukin and Dr. Jacob Gunther gave me lots of valuable advices. The discussions with them really helped my research. Hereby I also appreciate my other committee members and past committee members, Dr. Brandon Eames, Dr. Matthew Berkemeier, and Dr. Nick Flann.

I want to sincerely thank all the people who had ever helped the MAS-net project, including the advisors, Dr. Kevin Moore and Dr. YangQuan Chen; the team members, PengYu Chen, ZhongMin Wang, Anisha Arora, HaiYang Chao and William Burgeous; the CSOIS members, Dr. Lili Ma, Dr. JinSong Liang, Dan Storment; Especially, I want to thank PengYu Chen for his significant contributions to the MAS-net project. Without him, the achievement is not possible.

In the past several years, I have learned much from the discussions or collaborations with my colleagues. Hereby I present my appreciation to them. Nazif Cihan Tas, XiaoChun Xu, Vehbi Cagri Gungor. I am grateful for the various of help from my friends, Dr. Hyo-Sung Ahn, Dr. Dong Wei, Peng Zhao, Tian Xia, Michael Frashinger, Stanislava Soro.

Zhen Song

Contents

List of Tables

List of Figures

Chapter 1
Introduction

1.1 Background And Literature Review

1.1.1 What is Sensor Network

Sensor network [2] is considered as "one of the important technologies of the 21st century" [3]. In general, sensor network is referring to sensors that has been connected by computer networks. The sensors measure physical quantities of the world, while computer networks integrate and interpret sensor data.

This concept was proposed decades ago. For example, it is common to see networked data acquisition (DAQ) systems, such as Labview systems [4], in industry. However, many researchers have recognized that the modern computer, sensor and communication technologies has significantly changed the sensor networks.

Since microprocessors are very affordable now, sensor data can be processed by local processors before being transmitted to the base station, or the sink. The concept of "smart sensor" can be realized recently.

Communication technologies, either wired or wireless, are significantly improved in the past several years. Wireless technologies are especially under fast developments. More and more new communication technologies have been developed. WiFi, Bluetooth, CDMA, UWB, etc. Comparing to the past, it is much easier to connect thousands of sensor nodes with the help of those new technologies.

Due to these new enabling technologies, the methods to deploy sensors and collect the sensor data are also different from before. In general, those new technologies have potentials to reduce the cost of sensor networks, improve the energy efficiency, network scalability and reliability. How to properly use those new technologies is an active research topic.

The sensor networks can be classified into different categories. Among them, wireless sensor network (WSN) [5, 6] and distributed sensor network (DSN) [7] are among the two most important categories of sensor networks. This dissertation focuses on WSNs.

1.1.2 Introduction to WSN

WSNs [5, 6, 8] are not just traditional sensor networks with wireless communications. In fact, the philosophy of WSNs is different from other sensor networks in the past. The deployment methods, power consumption requirements and network protocols for WSNs are different than those wired sensor networks.

Currently, the computer and communication technologies have arrived in a stage where the hardware cost is so low that massive dense deployment of the wireless sensor nodes is affordable. A densely deployed WSN provides more samples on the physical world and improves the observation. However, the costs of replacing batteries for large scale sensor networks are significant. It is very important to improve the network life time by various of energy saving methods. The energy efficient protocols of WSN are under active research. Comparing to generic wireless communication protocols, the energy efficient protocols of WSN commonly uses knowledge on the structure of the sensor data to improve the performance. For example, data-centric routing approach [9] is proposed to replace IP-based routing: the route is computed based on the relative position between the data and the sensors, instead of IP addresses of the sensors. In addition, due to the nature of wireless communication, the topology of the network is not statics. The wireless communication protocols must adjust the route accordingly [10].

The future of WSN may come to the scenario of "smart dust" [11]: each sensor node is very small, even close to the size of dust. On the sensor nodes, microprocessors, sensors, communication devices, or energy harvesting apparatus are fully integrated. To explore the capability of the sensors, the sensor nodes are deployed in massive scales at high densities. Comparing to traditional sensor networks, the WSNs are in larger scale and normally work in distributed fashion.

The following items are some common features of wireless sensor networks:

- The WSN protocols support large scale networks. For example, IEEE 802.15.4 standard [12] is used in some sensor network products. The standard supports 2^{32} sensor nodes per network, in theory. In the literature of sensor network protocol design, it is common to see that a certain algorithm is tested on a network with tens of sensor nodes [10, 13] and simulated with hundreds or thousands of nodes [7, 14]. The sizes of WSNs are in larger scales comparing to most of other wireless communication protocols. For example, the Bluetooth standard was designed for Wireless Personal Area Network (WPAN) in stead of sensor network. The Bluetooth standard only supports 8 devices per network [15].

- The costs of wireless communication are relatively low, in terms of money, energy and computation. For example, the maximum power consumption of Telos mote, an example of WSN sensor node, is in the level of tens milliwatts and the communication stack is small enough for low cost microprocessors [16]. The price of that Telos sensor node is around $70 at the moment. As a comparison, WiFi (IEEE 802.11) devices are normally powered by wires, instead of by common AA batteries, because their energy consumption is much higher. In addition, WiFi devices are normally more costly and have larger form factors.

- The sensor nodes have certain computing capabilities. Sensor data are normally pre-processed on the sensor nodes. The pre-processing could be data filtering, compression, or encryption on the information of interest. For example, the sensor data may be stored on a local database [17] and the node only delivers the data that are queried by the base station. Taking the advantages of the on-board processing power the valuable communication resources are used for efficiently.

- The sensor nodes are normally deployed densely. Usually, dense deployment is more preferable as far as it is affordable. A denser deployment provides better estimation resolution on the physical world.

- Sensor networks are normally supported by specially designed communication hardware or protocols. Due to the difference between sensor data and generic communicate data, such as speech or video, the generic communication systems are not the most efficient approach for sensor data transportation. For example, instead of pursuing high communication speed, as many other communication standards do, the IEEE 802.15.4 standard employs low rate communication. Since high speed is not required by many sensor systems, adopting low rate communication is a better strategy, in terms of cost, size and energy requirements, etc.

WSN can be used in a wide variety of remote monitoring and control applications ranged from environmental and human body monitoring [18] to military surveillance [19–21], building automation [22], industrial monitoring and control [23], homeland security [24], air pollution detection [25], wild fire monitoring [26], detection of persons and vehicles in open areas [27], and reconnaissance tasks [5], etc.. In typical remote monitoring applications, sensor nodes are deployed in an ad hoc manner over an area of interest. Individual sensor nodes can measure physical quantities and communicate sensing information to other sensors or to a sink (base station) by radio. Furthermore, sensor nodes have a limited ability to process information on an on-board CPU, and can store that information in memory. This is the reason why such wireless sensors are sometimes referred to as smart sensors or smart dust. The on-sensor processing and on-sink processing should cooperatively interpret sensor data to observe environments in energy efficient approaches. Although each individual node has limited capability, several such nodes can cooperate to accomplish complex tasks. To put it succinctly, WSNs provide the ability to connect the physical world to enterprize computing systems, thereby improving business processes and facilitating efficient decision making.

1.1.3 Research on WSNs from Different Aspects

As an interdisciplinary topic, WSNs have been studied from many different aspects by researchers from different backgrounds. Because the interpretation on the sensor data

and the structure of the sensor data are application-dependent, to our best knowledge, there is no unified theory that integrates all the sensor network design problems into one framework. Most of the research in sensor network are conducted under the guidance of one of the following aspects:

- Communication and networking aspect: how to send information by WSNs.
- Signal and system aspect: what information are useful and should be sent.
- Data and service aspect: how to store and query data.

Hereby we classify and review the current literatures based on the proposed classification methods.

1.1.3.1 Communication and networking

WSNs can be considered as networked communication systems. From the aspect of networking, new protocols or communication devices can be designed, in order to transport the input data to the sink and satisfies certain quality of services (QoS). Some examples of commonly considered QoS include network throughput, packet reception rate, delay, etc.

In the comprehensive survey for the protocols of sensor networks [2], the differences between WSNs and traditional ad hoc networks are summarized as the follows:

- WSNs are deployed in much larger numbers with higher densities, and prone to failures.
- The topologies of WSNs are frequently changing.
- Sensor nodes have limited capabilities of power, computation and memory.
- Each sensor node may not have an unique ID.

WSNs are also compared with the Internet [8, 28].

- Self-organizing is more important for WSNs.

- The routing method used in the Internet is not practical for WSNs. Instead, routing methods based on sensor location or sensor values may be more effective.

- Constraints on bandwidth and energy are more limited for WSNs.

Traditionally, computer networks are implement by different layers. The classical TCP/IP model has 5 layers. They are physical layer, data link layer, network layer, transport layer, and application layer. Most of the layers have their counterparts in WSN protocols.

On the physical layer, the central task is to develop low-cost and energy-efficient radios, on which both the academia and industry are keep proposing new solutions [12, 29–32]. On the data link layer, different Media Access Control (MAC) protocols are proposed. Some are developed for general sensor networks [33] and some are designed for specific application [14]. The network layer is responsible for network routing. Common routing metrics include the number of hops, RSSI, and link quality, etc. [34].

This layered network communication model is widely adopted. However, since it was developed for generic communication, it may not satisfy the requirements of WSNs. As paper [35] observed, a key challenge of the current sensor network systems is the lack of a general communication model. Many protocols behave well with them alone, but does not cooperate with each other. In fact, the layered model is not the only solution. For example, hybrid designs do not follow the layered model [2, 35]. In the hybrid models, one or several modules are operating cross several layers.

In summary, WSNs can be considered as a new type of computer networks. In addition to pursuit classical network performances, such as high throughput, high packet reception rate (PRR), and low message delivery delay, some new criteria, such as energy efficiency, fault tolerance and scalability, should also be considered. A good network model should be able to balance different performances and trade off the costs, in order to satisfy individual applications as much as possible.

1.1.3.2 Signal and system

WSNs are commonly used to observe physical systems. The sensor measurements can be considered as signals, while the network is an imperfect communication channel, which introduces delays and packet drops, etc. Thus, there are many challenges on signal processing and system identifications for WSNs. The follows are some examples.

- Sensor calibration: Unlike the calibration for a single-sensor system, calibration on sensor networks is more complex. Since there are normally a large number of sensors in the network, it is normally impossible to manually calibrate sensors one by one. Automatic and systematic calibration methods are required [36, 37].

- Sensor selection and placement: Due to the properties of the physical systems, the "quality" of the sensor data is position-dependent. In order to save the communication energy, we need to ensure that just enough "good" data are sent through the communication channels; no "bad" data or "more than enough" data should be transmitted. It is important to select sensors or positions of the sensors based on the quality of their sensing data [14, 38–40].

- Mobility of sensor node: Mobile sensor nodes are feasible in the context of WSNs. Some methods are proposed to take the advantages of the mobility of sensor nodes for better sensing [41–44].

- Detection and estimation: It is desirable to have distributed detection or estimation algorithms, in order to enhance the scalability and fault tolerance of WSNs. For example, distributed regression [45], distributed least squares fitting [46] and other distributed algorithms [25, 47] are discussed.

- Target tracking: As an extension to the classic target tracking problem, energy efficient or distributed tracking are discussed [48–51], under the context of WSNs.

Of course, there are much overlapping on the above topics. For example, Aranda et al take the advantages of the mobility of robotic wireless sensor nodes to track targets [51].

The knowledge on the physical model of the system under observation may significantly improve estimation precision and reduce the energy cost. For example, we may want to deploy a WSN-based predictive maintenance system on an expensive engine [23] to predict possible failures in future. If the model of the engine is known, some important questions can be answered. By analyzing the physical model of the engine, we can determine the observability of an internal physical quantity, e.g., an internal state. If it is not observable, then the quantity can not be estimated, no matter how many sensors are installed. Based on the model, the sensor deployment can be optimized. For example, one approach for the optimal sensor placement problem is to formulate it as a convex optimization problem. The problem can be solved under the framework of optimal experiment design [44, 52, 53].

1.1.3.3 Data and service

Wireless sensor networks can be considered as distributed database systems. The users may query the data that have been collected by the sensor nodes. Due to the unique properties of the WSNs, the queries on WSNs are not the same as the queries on standard databases.

For WSNs, the users are interested in high level information, and data aggregations are usually required. For example, one may query "the averaged temperature of the 4th floor," instead of the measurements on temperature values of each sensor. One method to implement the above query is to aggregate the sensor data at sink [17].

Although the query on WSN can be considered as a special routing problem [54] and implemented by the standard address-centric routing, this approach may not be efficient, since the raw sensor data have much redundant information. It may be more efficient to transmit raw data by the address-centric routing methods, where the sensor data may be aggregated during the transmission. The route may be affected by the sensor data [9].

1.2 Dissertation Motivation and Application Scenarios

1.2.1 Motivation and Scenarios of MAS-net

Chapter 2 is motivated by our project named MAS-net, which stands for mobile actuator-sensor networks (MAS-net) [55–60]. This project is proposed to combine the latest sensor network technologies with robotics technologies for an application-oriented high-level tasks, namely, characterize, estimate and control diffusion process by networked mobile actuators and sensors.

The future working scenario of the MAS-net system is presented in [55, 61]. Hereby we review it in brief. The scenario is shown in Fig. 1.1. The numbers of the following comments are associated with the plots with the same numbers in Fig. 1.1.

1. In the middle of a city, terrorists release a plume of chemical, biological, radiological (CBR) fog to the air. The diffusion of the plume is influenced by the city building structures and the air flow. The poisonous plume is detected by a static sensor network.

2. A group of Unmanned Aerial Vehicles (UAVs) receive commands from a base station to estimate the boundary of the dangerous plume. Equipped with proper sensors and wireless communication modules, these UAVs make up an ad hoc wireless sensor network.

3. UAVs fly toward the plume and transmit their sensor data back to the base station in real-time.

4. Initially, the plume estimation program running at the base station does not have the full knowledge on the parameters of the plume diffusion model. While getting more and more information from the sensors, the estimated parameters converge to the true values, and the plume boundary prediction becomes more and more precise. Meanwhile, the base station sends commands to direct the UAVs to the estimated future boundary area, where they gather new information.

5. Once the CBR plume is satisfactorily defined, the UAVs are redirected to the appropriate locations to release proper anti-agents to eliminate the plume.

6. The plume is eliminated within the minimum time (or satisfying other optimization constraints). The city becomes safe again.

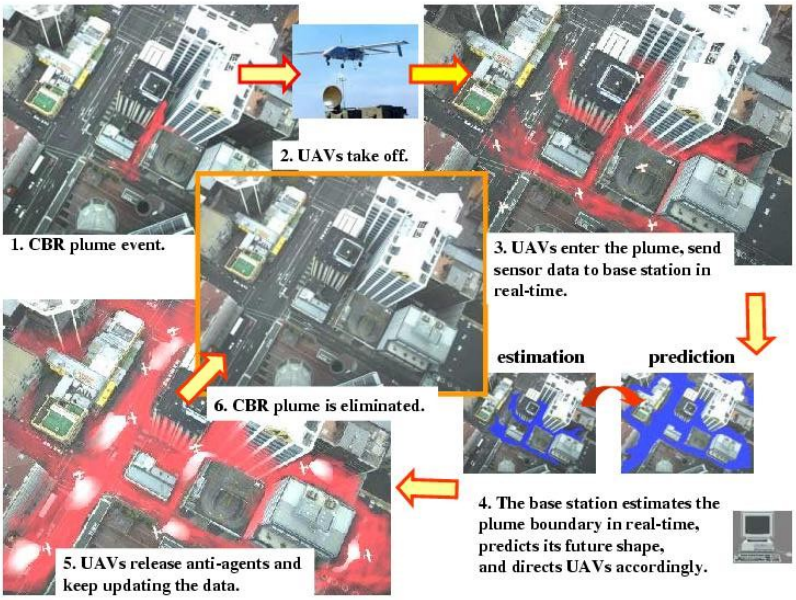


Fig. 1.1: A typical working scenario for mobile actuator-sensor network.

The MAS-net problem is challenging. Thus, a simplified problem is discussed in the dissertation. In the problem, differentially-driven robots equipped with sensors measure a diffusing fog. The fog is within a flat container with a transparent cover. Since the height of the container is not much, the diffusion can be considered within a 2D domain, instead of 3D. In the current stage, the motivation is to observe the fog by those mobile robots.

The configuration of the MAS-net testbed is shown in Fig. 1.2. A program called Integrated Control System (ICS) is executing on a PC named base station. From the Graphical User Interface (GUI) of the ICS, users can control the MAS-net testbed. The base station is connected to a wireless sensor node, MICA2 board, through the programming board. The base station communicates with the so called MASmote [58] robots via this sensor node. The MASmotes are palm-size differentially driven robots that equipped with fog sensors and wireless sensor boards, which can communicate with the base station

and other MASmotes. The fog sensors estimate the concentration of the fog underneath the transparent cover, on top of which the MASmotes maneuver. A camera is hung on top of the testbed and connected to the base station. Based on the video stream from the camera, the ICS on the base state detects the positions and orientations of the MASmote robots according to the unique markers on top of each robot.

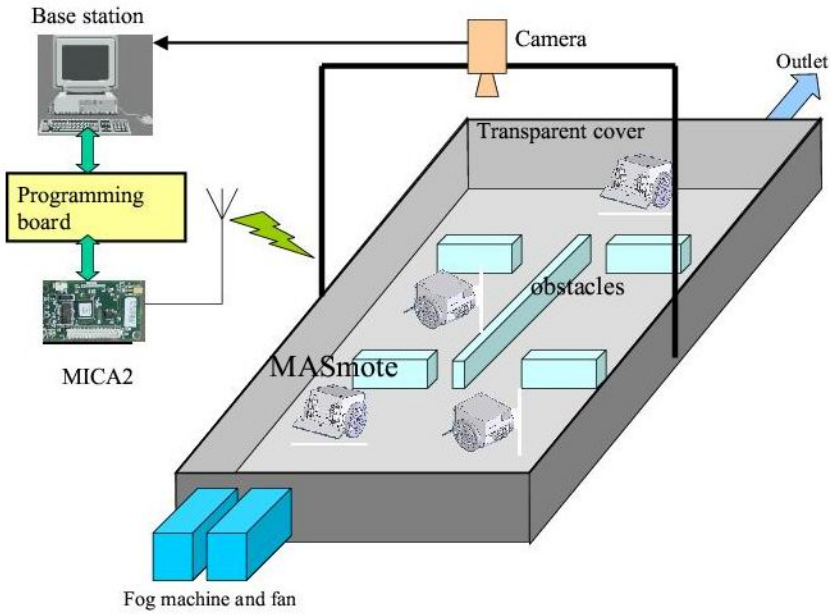


Fig. 1.2: 2D testbed configuration for MAS-net project.

The fog is generated by a stage fog machine and diffuses under the cover. An electrical fun is placed close to the fog machine to simulate the wind. To simulate city structures, some obstacles are placed under the cover to provide complex boundary conditions that similar to engineering practices. A picture of the testbed is shown in Fig. 1.3. In this picture, the markers are not placed on top of the robots. The MASmote robots with markers are shown in Fig. 1.4.

The pictures in Fig. 1.4 are captured from our movie [62]. The movie is merged from two video streams. A video from camcorder (bottom left) is overlapped on a video (background) from the GUI of the ICS. This movie demonstrates simple collaborative fog estimation. In this demo, a white paper board is used to simulate the fog because it is static and it is easy to verify the correctness of robots' behaviors. Once they enter the "foggy"

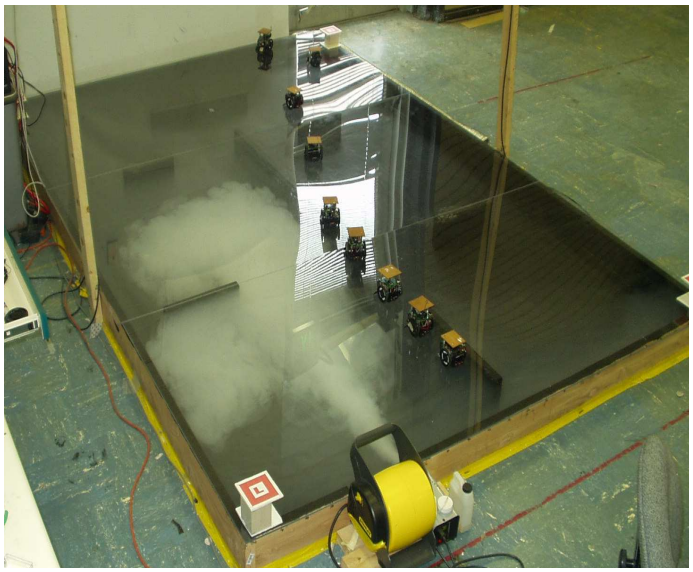


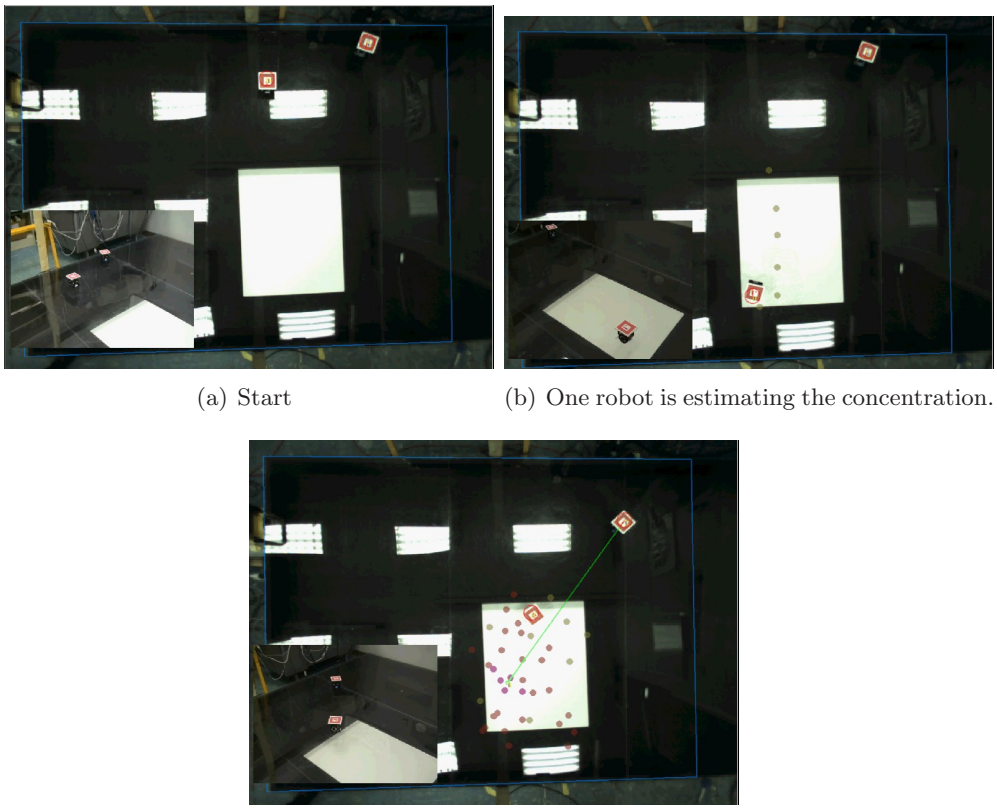
Fig. 1.3: The MAS-net testbed.

area, the robots should wander inside the area and report the simulated fog concentration to the ICS on the based station. Those dots on the screen are the augmented reality images that present the concentration of the "fog." After the first robot found a plum large enough, it sends a "help me" message to the other robots and guide them into the area with fog.

In Chapter 2, a problem motivated by the MAS-net project is discussed. Given several differentially driven robots, how to determine the optimal trajectories, such that the observation on the parameters of a diffusion fog is optimized? Since the fog is a distributed parameter system (DPS) and modeled by partial differential equations, mathematically speaking, the problem is as the follows: Find an optimal control law for sensors with nonholonomic constraints to observe parameters of a DPS. A numerical method is proposed and studied in Chapter 2.

1.2.2 Scenarios of Sensor Selection

Chapter 3 focuses on the sensor selection problem. As aforementioned, energy conservation is a key issue for WSNs. When WSNs are involved in real-time observation tasks, such as environment monitoring or target tracking, some sensors must stay in active mode



(a) Start (b) One robot is estimating the concentration.

(c) The other robot is called for help.

Fig. 1.4: Cooperative fog estimation.

and submit their measurements to the sink (base station) periodically. Of course, the less number of selected sensors the better, provided the observation error is small enough. The problem of sensor selection is to choose the proper sensors and just enough number of sensors such that high precision estimation is achieved with the least energy costs.

In WSN applications such as building automation, an event of interest could be a fire in some area of a building; it could also be the leakage of toxic gases in certain region. However, the exact location at which such an event takes place is not known. Moreover, the sensors in the vicinity of the event in question measure physical parameters like temperature, but they may not by themselves establish the location of the event. It is even possible that no physical sensor could measure the event's location directly. Thus, in order to take any actuating action like turning on the sprinklers, it is first required to accurately

establish the location of the event. The location of the event is thus an unknown quantity that should be estimated based on sensor data.

Once an event in question is detected by one or more sensor nodes and its location is established, the sensors communicate their measured readings to a sink to facilitate actuating actions. However, in order to conserve the limited energy budget on each sensor node, and also to optimally utilize limited radio bandwidth, it is not efficient to have every sensor that detected the event to communicate its readings at the maximum possible transmission rate. This will result in not only rapid depletion of on-board battery power on each node, but also lead to severe network congestion resulting in loss of valuable communication packets pertaining to the event. Intuitively, data from assigning a high transmission rate to a sensor far away from the fire fetches only limited information. Those samples have more noise and the quality of the information is not satisfied. It is wise to increase the transmission rate for those sensors that are closer to the fire, since they can provide "good" data.

The sensors with high transmission rates are selected and those with low transmission rates are called unselected. The sensors that are far from the fire should not be selected. This does not mean that every sensor that is "close," e.g., within some threshold distance, to the event in question is guaranteed to be selected. We need a systematic, analytic approach to select sensors. In fact, the results in Chapter 3 indicate that the closest sensors are not always selected. It is also proved in the chapter that only a small number of sensors are required to be selected in order to achieve the optimal precision. Those conclusions may be counter intuitive. The detailed analysis and experiment results are presented in Chapter 3.

1.2.3 Scenario for WSN-based Localization

Localization is a fundamental function for WSNs [63], as well as many other applications, such as mobile robots [64, 65], navigation [66], etc.

Currently, global positioning system (GPS) probability is well known localization technology the public. Despite of the achievements of GPS, the GPS is not ideal or available

for many applications. For example, GPS signals may be blocked by buildings, heavy foliage, large metal objects or strong electrical field [67]. In addition, the costs and energy requirements of GPS receivers may be not acceptable for some applications [68]. GPS is not available for indoor localization, since the satellite signals of the GPS system are blocked by building structures. For outdoor application, although GPS is an option, it is mainly complimentary to the WSN-based localization systems rather than competitive. In typically WSN deployments, a large number of sensor nodes should be equipped with low-cost, energy efficient localization devices that designed for WSNs [31, 69]. The GPS equipments are only installed on several beacon nodes (anchor nodes) to provide global positions.

Comparing to the localization technologies for robotics and navigation, the WSN-based localization is different. Common sensors used for robot localization and navigation, such as sonar or laser scanners, are too costly for WSNs, in terms of money, energy cost and physical size.

Although the importance of WSN-based localization has be recognized, and active research has been conducted for years, the technology is still not matured in the sense that no out-of-the-shelf WSN localization systems are available today. As a brief summary, we list the unique constraints or challenges of the WSN localization problem.

- To fit into the strategy of the WSNs, the localization hardware for sensor nodes must be portable, low cost, energy efficient and precise enough. So far, no commercial hardware solution satisfies all those requirements.

- Since the sensor nodes are commonly deployed in ad hoc approaches, the rough estimates on the positions of the sensor nodes may not be available. If the localization algorithm is based on the common least squares (LS) approach, the position estimates may not converge to the real values, since the randomly generated initial positions may not be close enough to the global minimum, which is the true value of the sensor's position [70]. In addition, since both of the sensor nodes (whose positions

are unknown) and the beacon nodes (whose positions are known) may be placed in ad hoc manners, the positioning on some sensor node may be ambiguous since not enough beacons' signals may be received [71].

- It is common to progressively deploy the beacon nodes. If the batteries on certain beacons are depleted or the positioning errors are not satisfied, more beacon nodes should be added. In fact, adaptive progressive beacon placement is discussed under the context of WSN [72].

- Some sensor nodes in a WSN may be mobile. This is yet another challenge for localization.

- Multi-hop is common for WSNs. A beacon node may not be able to directly communicate with a sensor node. Some WSN localization methods, e.g., the DV-hop algorithm [72], locate sensors by sensor networks with multi-hops.

1.3 Summary of Dissertation Contributions

In the dissertation, several WSN problems are unified under the framework of optimal experimental design. The essential contributions are as the follows:

- Propose a numerical method to optimize the trajectories of mobile sensor nodes to estimate parameters of DPSs.

- Propose a sensor selection method to select the minimum number of sensors with the least communication energy costs for the optimal parameter estimation.

- Prove the existence of a class of implicit optimal sensor selection methods. The proof also provides guidance on the design of future sensor selection methods as well as the parameter tuning of those methods.

- The robustness and performances of the sensor selection algorithm have be verified by extensive hardware experiments and simulations.

- Propose an asynchronous time difference of arrival (TDOA) localization method for energy efficient localization by WSNs.

- Based on the TDOA method, develop a method to optimize the beacon placement for robust localization.

1.4 Dissertation Organization

The organization of the dissertation is as the follows. The mobile sensor trajectory optimization problem is discussed in Chapter 2, based on the context of MAS-net project. Chapter 3 focuses on the sensor selection problem. Chapter 4 presents the research on localization and beacon placement problems. Chapter 5 concludes the dissertation. Finally, notations and some comments on the implementations are presented in the Appendices.

Chapter 2
Mobile Sensor Trajectory Optimization for Distributed Parameter System Identification

2.1 Motivation and the Application Scenario

Thanks to the advances of the technologies, large scale WSNs or mobile WSNs are more affordable. Different applications could take different advantages of WSNs. For WSNs used for monitoring lumped parameter systems, such as an electrical motors, the large number of sensors introduce redundance for the measurement, thus enhance the robustness to faulty sensors. However, this is not the only application scenario of WSNs. There exists a wide class of processes whose behaviors are described by Partial Differential Equations (PDEs) due to the inherent spatial and temporal variabilities of their states. They are commonly termed the Distributed Parameter Systems (DPSs). Since a DPS has infinite number of states, large scale WSNs may provide fine grid estimates with better precision as comparing to the traditional wired sensing systems, which are normally in much smaller scale due to the difficulties of wired connections. These scenarios may result to many useful applications of WSNs as the follows:

1. Wild fire monitoring [26].
2. Landslide prediction [73].
3. Volcano status monitoring [74].
4. Diffusive pollution monitoring and control [56, 57].
5. Water pollution monitoring [19].
6. Chemical plume tracking [75].

2.2 System Identification for DPS

Currently, DPS occupies an important place in control and systems theories [76–81]. One of the basic and most important questions in DPSs is parameter estimation, which refers to the determination from observed data of unknown parameters in the system model such that the predicted response of the model is close, in some well-defined sense, to the process observations. For that purpose, the system behavior or response is observed with the aid of some suitable collection of discrete sensors, which reside at predefined spatial locations. However, the resulting measurements are incomplete in the sense that the entire spatial state profile is not available. Moreover, the measurements are inexact by virtue of inherent errors of measurement associated with transducing elements and also because of the measurement environment. These factors lead to the question of where to locate sensors so that the information content of the resulting outputs with respect to the distributed state and PDE model be as high as possible.

It is widely accepted that making use of sensors placed in an "intelligent" manner may lead to dramatic gains in the achievable accuracy of the resulting parameter estimates, so efficient sensor location strategies are highly desirable. In turn, the complexity of the sensor location problem implies that there are a very few sensor placement methods, which are readily applicable to practical situations. What is more, they are not well known among researchers. This generates a keen interest in the potential results, as the motivations to study the sensor location problem stem from practical engineering issues. Optimization of air quality monitoring networks is among the most interesting ones. One of the tasks of environmental protection systems is to provide expected levels of pollutant concentrations. To produce such a forecast, a smog prediction model is necessary, which is usually chosen in the form of an advection-diffusion partial-differential equation. As more sensor measurements un-avoidably introduce more energy costs and hereby increase the maintenance budget, we are faced with the problem of how to optimize their locations to obtain the most precise model with a limited number of sensors. Other stimulating applications include, among other things, groundwater modeling, recovery of valuable minerals and hydrocarbon from underground permeable reservoirs, gathering measurement data for calibration

of mathematical models used in meteorology and oceanography, automated inspection in static and active hazardous environments where the trial-and-error sensor planning cannot be used (e.g. in nuclear power plants), or emerging smart material systems.

The sensor placement problem was attacked from various angles, but the results communicated by most authors are limited to the selection of stationary sensor positions [52, 82, 83]. An intuitively clear generalization is to apply sensors, which are capable of continuously tracking points providing at a given time moment best information about the parameters (such a strategy is usually called continuous scanning). However, communications in this field are rather limited. [84] considers the determinant of the Fisher Information Matrix (FIM) associated with the parameters to be estimated as a measure of the identification accuracy and looks for an optimal time-dependent measure, rather than for the trajectories themselves. The work [85] was intended as an attempt to properly formulate and solve the time-optimal problem for moving sensors, which observe the state of a DPS so as to estimate some of its parameters. Note that the idea of moving observations has also been applied in the context of state estimation [86–89], but those results can hardly be exploited in the framework considered here as those authors make extensive use of some specific features of the addressed problem (e.g. the linear dependence of the current state on the initial state for linear systems).

It should be emphasized that technological advances in communication systems and the growing ease in making small, low power and inexpensive mobile systems now make it feasible to deploy a group of networked vehicles in a number of environments [3, 19, 86, 90, 91]. A cooperated and scalable network of vehicles, each of them equipped with a single sensor, has the potential to substantially improve the performance of the observation systems. Applications in various fields of research are being developed and interesting ongoing projects include extensive experimentation based on testbeds. The problem to be discussed in this paper caught our attention while working on our MAS-net experimental platforms [55–60].

The MAS-net project is proposed to combine the latest sensor network technologies with mobile robotics for an application-oriented high-level task, namely, characterization,

estimation and control of an undesired diffusion process by networked mobile actuators and sensors. One potential solution is to estimate the parameters in a "closed-loop" or "on-line" approach, as mentioned in the last chapter of [92]. This idea can be explained as follows. With arbitrary initial values of the unknown parameters, the system starts to drive sensors in an "optimal" trajectory with respect to those parameters. Sensor data are then collected while the sensors are moving. On the basis of the collected data, parameter estimates are improved and the moving sensor trajectories are then updated accordingly. Then, the sensors are driven to follow the newly updated trajectories based on the parameters estimated. Through this "closed-loop" iteration or the recursive online adaptation, the estimated parameters converge to the true values of the DPS. This so-called "online" mode was listed as one of the important future research efforts.

In this paper, we focus on the "control for sensing" part, that is, given an estimate of the DPS parameters, how to drive the mobile sensors optimally in the sense that the effect of the sensor noise can be minimized. We present a numerical solution for a mobile sensor motion trajectory scheduling problem under non-holonomic constraints as in MAS-motes [58], the two-wheeled differentially-driven mobile robots, in our MAS-net project. More details of the project is presented in Chapter 1.

From the theoretical aspect, the key challenge of the project is to develop real-time parameter estimation and state estimation of a class of distributed parameter systems (DPSs) by a swarm of mobile sensors with nonholonomic constraints and limited communication capability. In addition, mobile actuators (e.g., a mobile robot equipped with a chemical neutralizer dispenser) with the same nonholonomic constraints will be added to control the DPS (basically, to reduce the concentration) with the help of the mobile sensors.

While observing a DPS, it is most often impossible to measure the system states over an entire spatial domain, and therefore the problem of where to locate the measurement sensors becomes very important. It is not trivial to estimate the parameters or the states of the DPS by a limited number of sensors. The study of the DPS identification problem

started about 40 years ago, but the number of papers that discussed the sensor-motion-scheduling problem is still limited. Many of them discussed the optimal-placement problem for static sensors.

The model-based adaptive measurement and control problem of the MAS-net project is formulated in [56, 57]. To implement this distributed control system, the parameter estimation for the DPS is required, and the choice of best experimental conditions for that purpose is referred to as an "optimum experimental design" problem as discussed in [93–96].

The recent publications [52,92] are closely related to the MAS-net estimation problem. In [52,92], the dynamic-sensor-motion scheduling problem is studied intensively with many practical considerations such as robust design, collision avoidance, etc.

The rest of this chapter is organized as follows. The formulation of the MAS-net estimation problem is described in Section 2.3, in which the dynamic model for differential-drive mobile robots is presented in Section 2.3.1 and the objective function for the optimal sensor motion scheduling is described in Section 2.3.3. Section 2.3.4 reformulates the problem in the framework of optimal control. In Section 2.4, a numerical solution procedure for this problem is presented. A Matlab optimal control toolbox RIOTS is briefly described in Section 2.4.1 and Section 2.4.2 describes a method to incorporate the Matlab PDE Toolbox [97] and the RIOTS, cf. Section 2.4.1. Some illustrative simulation results are presented in Section 2.5 with remarks on the obtained results. Section 2.6 concludes this chapter. Further comments on the implementation of the simulation are presented in the Appendix.

2.3 Problem Formulation of the Sensor-Motion Scheduling for Diffusion Systems

In this section, the model of our diffusion system and the model of our differential-drive robots are presented in Section 2.3.1 and Section 2.3.2, respectively. After introducing a class of objective functions in Section 2.3.3, the MAS-net estimation problem is reformulated in the framework of optimal control, and ready to be solved by RIOTS.

2.3.1 The Dynamic Model of Differentially-Driven Robots

MASmote [58] is a differentially-driven ground mobile robot as illustrated in Fig. 2.1. Its dynamic model can be described by (2.1), where the symbols are defined as follows:

- m: the weight of the robot.
- I: the inertia of the robot along the z axis. Note that I is a scalar. In the literature, $\mathcal{M}$ can be a 3 by 3 inertia matrix of the robot. The I in (2.1) is the entry at the 3rd row and 3rd column of $\mathcal{M}$, i.e. $\mathcal{M}_{(3,3)}$.
- l: the length of the robot's axis.
- r: wheel radius. The left and right wheels have the same radius.
- b: the edge length of the robot's square chassis. It is assumed that the wheels and the axis are mounted on the square chassis.
- α: the yaw angle as shown in Fig. 2.1.
- (x, y): the coordinate of the center of the axis.
- τ_l,τ_r: the torque applied on the left and right wheel, respectively.

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\alpha} \end{bmatrix} + \begin{bmatrix} 2b & 0 & 0 \\ 0 & 2b & 0 \\ 0 & 0 & bl^2/2 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} r\cos(\alpha) & r\cos(\alpha) \\ r\sin(\alpha) & r\sin(\alpha) \\ -rl/2 & rl/2 \end{bmatrix} \begin{bmatrix} \tau_l \\ \tau_r \end{bmatrix} \tag{2.1}$$

In (2.1), the mobile robot is represented in a form of a 2nd order system. For convenience, the corresponding state space form can be easily derived by introducing $\mathbf{x}$, the extended system state vector defined as $\mathbf{x} := [x\ y\ \alpha\ \dot{x}\ \dot{y}\ \dot{\alpha}]^T$, and τ is defined as

$$\tau = \begin{bmatrix} \tau_l \\ \tau_r \end{bmatrix}.$$

Note that $\mathbf{x} \neq x$. $\mathbf{x}$ is the state vector, while x is the robot's position on the x-axis.

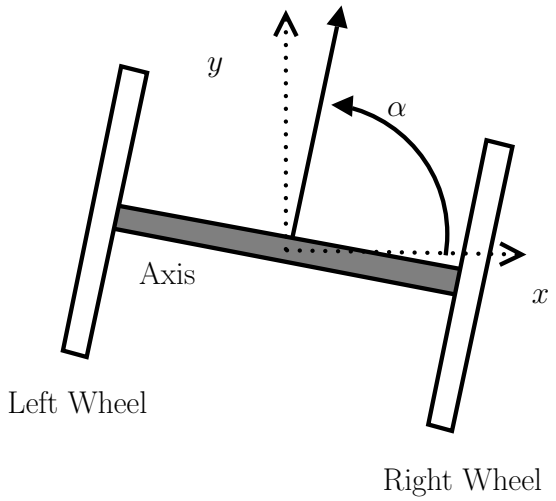


Fig. 2.1: A differentially-driven mobile robot.

To have a compact notation, let us define matrix A and B as

$$A := \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -2b/m & 0 & 0 \\ 0 & 0 & 0 & 0 & -2b/m & 0 \\ 0 & 0 & 0 & 0 & 0 & -bl^2/(2I) \end{bmatrix},$$

and

$$B := \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ r\cos(\alpha)/m & r\cos(\alpha)/m \\ r\sin(\alpha)/m & r\sin(\alpha)/m \\ -rl/(2I) & rl/(2I) \end{bmatrix}.$$

Thus, the robot dynamics can be written as

$$\dot{\mathbf{x}} = A\mathbf{x} + B\tau. \tag{2.2}$$

Note that B depends on $\mathbf{x}$.

To solve the multi-robot-motion-scheduling problems in Section 2.5, we need to write the dynamics of three robots as a single dynamic system. Denote the states of each robot

in (2.2) as $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$, respectively. After defining

$$\mathbf{x}_T := \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}, \quad A_T = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & A_2 & 0 \\ 0 & 0 & A_3 \end{bmatrix},$$

$$B_T = \begin{bmatrix} B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \end{bmatrix}, \text{ and } \tau_T = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix},$$

where A_j, B_j are for the j-th robot, the dynamics of all three robots can be written compactly as follows:

$$\dot{\mathbf{x}}_T = A_T \mathbf{x}_T + B_T \tau_T. \tag{2.3}$$

2.3.2 The Model of the Diffusion Process

For comparison purposes, here we use the same diffusion system model as in Example 4.1 in [52]. We rewrite it using our notation in the following form:

$$\begin{aligned} \frac{\partial u(x,y,t)}{\partial t} &= \frac{\partial}{\partial x}\left(\kappa(x,y)\frac{\partial u(x,y,t)}{\partial x}\right) \\ &+ \frac{\partial}{\partial y}\left(\kappa(x,y)\frac{\partial u(x,y,t)}{\partial y}\right) \\ &+ 20\exp(-50(x-t)^2), \\ &\quad (x,y) \in \Omega = (0,1)\times(0,1), t \in T, \\ u(x,y,0) &= 0, \ \ (x,y) \in \Omega, \\ u(x,y,t) &= 0, \ \ (x,y,t) \in \partial\Omega \times T, \\ T &:= \{t | t \in (0,1)\}, \\ \kappa(x,y) &= c_1 + c_2 x + c_3 y, \\ &\quad c_1 = 0.1, \ c_2 = -0.05, \ c_3 = 0.2, \end{aligned}$$

where $u(x,y,t)$ is the concentration, (x,y) is the spatial coordinate, c_1, c_2, c_3 are the nominal parameters, and t is the time.

2.3.3 The Objective Function for Sensor-Motion Scheduling

In this paper, the aim of the optimization is to minimize the sensor noise effect. For the i-th mobile sensor, its observation is assumed as follows:

$$z_i(t) = u(\mathbf{x}_i(t), t) + \epsilon(\mathbf{x}_i(t), t),$$

where ϵ is white noise with statistics

$$E\{\epsilon(x,y,t)\} = 0,$$

$$E\{\epsilon(x,y,t)\epsilon(x^*,y^*,t^*)\} = \sigma^2\delta(x-x^*)\delta(y-y^*)\delta(t-t^*).$$

The positions are in the domain of the diffusion process, i.e. $(x,y) \in \Omega$ and $(x^*,y^*) \in \Omega$. The δ is Dirac's delta function, and σ is a positive constant.

The objective function is chosen to be the so-called D-optimum design criterion defined on the Fisher Information Matrix (FIM), which is defined as the follows:

Definition 2.3.1 (Fisher Information Matrix) *If a measurable random variable X depends on parameter θ, and the likelihood function is $l(\theta;X)$. Then the Fisher information is*

$$\mathcal{I} = E\{[\frac{\partial}{\partial\theta}\log l(\theta;X)]^2\}$$

The matrix form is Fisher information matrix, which is M, with the i,jth entry defined as:

$$M_{(i,j)} = E\{\frac{\partial}{\partial\theta_i}\log l(\theta;X)\frac{\partial}{\partial\theta_j}\log l(\theta;X)\} \tag{2.4}$$

Remark An important application of FIM is to estimate the estimation error bound by the Cramér-Rao bound. The details are presented in Appendix B.

Up to a constant multiplier, the FIM constitutes the inverse of the covariance matrix for the least-squares estimator defined as the minimizer of the following "fit-to-data" criterion:

$$J_1(c) = \frac{1}{2}\int_T \|z(t) - \hat{u}(\mathbf{x},t;c)\|^2 \mathrm{d}t. \tag{2.5}$$

The notation $\hat{}$ in (2.5) indicates the predicted value. For n robots, $J_1(c)$ becomes

$$J_1(c) = \sum_{j=1}^{n}\frac{1}{2}\int_T \|z_j(t) - \hat{u}_j(\mathbf{x},t;c)\|^2 \mathrm{d}t.$$

Then, the FIM of n robots is defined as the follows:

$$M = \sum_{j=1}^{n}\int_0^{t_f}\left(\frac{\partial u(\mathbf{x}_j(t),t)}{\partial \mathbf{c}}\right)^T\left(\frac{\partial u(\mathbf{x}_j(t),t)}{\partial \mathbf{c}}\right)\mathrm{d}t, \tag{2.6}$$

where

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}.$$

The derivation from (2.4) to (2.6) is presented in Appendix B. Note that $\mathbf{x}_j$ is the state vector of the j-th robot. The Here $\mathbf{c}$ is the parameter vector in the DPS to be identified, and the partial derivatives are evaluated at $\mathbf{c} = \mathbf{c}_0$, a preliminary estimate of $\mathbf{c}$.

Note that the FIM, M, is a matrix. Thus, there are many metrics that can be defined to indicate the volume of the matrix. The D-optimality criterion used in this paper is defined as

$$\Psi(M) = -\ln\det(M).$$

Other optimality criteria are applicable but not discuss in this chapter. The comparisons among different criteria are presented in Chapter 3.

The objective function for the MAS-net estimation problem is to minimize $J_2(\mathbf{x}) = \Psi(M)$. Our goal here is to find the optimal control function $\tau \in L_\infty^{2n}[t_0, t_f]$ for n two wheel differentially-driven mobile sensors together with the initial states $\mathbf{x}(t_0) = \xi \in \mathbb{R}^K$ where $K = 6n$ and $t \in [t_0, t_f] = [0, 1]$, such that $J_2(\mathbf{x})$ is minimized.

2.3.4 Problem Reformulation in the Optimal Control Framework

According to the general optimal control problem formulation in RIOTS [98], our optimal mobile sensor motion scheduling problem can be formulated as follows:

$$\min_{(\tau,\xi)\in L_\infty^{2n}[t_0,t_f]\times\mathbb{R}^K} J(\tau, \xi) \tag{2.7}$$

where

$$J(\tau, \xi) = g_0(\xi, \mathbf{x}(t_f)) + \int_{t_0}^{t_f} l_o(t, \mathbf{x}, \tau)\mathrm{d}t$$

subject to the following conditions and constraints:

$$\begin{aligned}
&\dot{\mathbf{x}} = h(t, \mathbf{x}, \tau), \\
&\mathbf{x}(t_0) = \xi, \quad t \in [t_0, t_f], \\
&\tau_{j,\min}(t) \le \tau_j(t) \le \tau_{j,\max}(t),\ j = 1, \cdots, n, t \in [t_0, t_f], \\
&\xi_{j,\min}(t) \le \xi_j(t) \le \xi_{j,\max}(t),\ j = 1, \cdots, k, t \in [t_0, t_f], \\
&l_{ti}(t, \mathbf{x}(t), \tau(t)) \le 0, \quad t \in [t_0, t_f], \\
&g_{ei}(\xi, \mathbf{x}(t_f)) \le 0, \quad g_{ee}(\xi, \mathbf{x}(t_f)) = 0.
\end{aligned}$$

For our optimal motion scheduling problem, $\dot{\mathbf{x}} = h(t, \mathbf{x}, \tau) = A\mathbf{x} + B\tau$ for the single robot case and for three robot cases $\dot{\mathbf{x}}_T = h(t, \mathbf{x}_T, \tau_T) = A_T\mathbf{x}_T + B_T\tau_T$. Here, we define $l_0(\xi, \mathbf{x}(t_f)) = 0$ and $g_0(\xi, \mathbf{x}(t_f)) = \Psi(M)$ to simplify the numerical computation. This technique is called solving an "equivalent Mayer problem." To understand the equivalent Mayer problem, let us start from the definition of some new notation. $g(\mathbf{x}_i)$ is called the sensitivity function, where

$$g(\mathbf{x}_i, t) := \left(\frac{\partial u(\mathbf{x}_i, t)}{\partial \mathbf{c}}\right)^T.$$

Then, the FIM in (2.6) is

$$M = \sum_{j=1}^{n} \int_{t_0}^{t_f} g(\mathbf{x}_j(t), t) g^T(\mathbf{x}_j(t), t) \mathrm{d}t. \tag{2.8}$$

Define the Mayer states as

$$\chi_{(i,j)}(t) := \int_{t_0}^{t} \varpi_{(i,j)}(\tau) \mathrm{d}\tau. \tag{2.9}$$

where

$$\varpi_{(i,j)}(t) := \sum_{l=1}^{n} g_{(i)}(\mathbf{x}_l(t), t) g_{(j)}(\mathbf{x}_l(t), t).$$

Denote χ_{dl} the stack vector which stacks all the entries on the diagonal and below the diagonal of χ to a vector. For example, if χ is a 2 by 2 matrix,

$$\chi_{\mathrm{dl}} = \begin{pmatrix} \chi_{(1,1)} \\ \chi_{(2,1)} \\ \chi_{(2,2)} \end{pmatrix}.$$

Then, the extended Mayer state vector $\tilde{\mathbf{x}}$ can be expressed as

$$\tilde{\mathbf{x}} := \begin{bmatrix} \mathbf{x} \\ \chi_{\mathrm{dl}} \end{bmatrix}.$$

Comparing (2.9) and (2.8), one can easily observe the key point of this equivalent Mayer problem. That is, $\chi(t_f) = M$ and χ_{dl} contains all the information of M since M is symmetric. After replacing the extended state vector $\mathbf{x}$ with the extended Mayer vector $\tilde{\mathbf{x}}$, we can get M without explicit integration.

Thus, when considering the equivalent Mayer problem, the models used for RIOTS are as follows:

$$\begin{aligned} \dot{\tilde{\mathbf{x}}} &= \begin{bmatrix} A\mathbf{x} + B\tau \\ \varpi_{\mathrm{dl}} \end{bmatrix}, \\ \dot{\tilde{\mathbf{x}}}_T &= \begin{bmatrix} A_T\mathbf{x} + B_T\tau_T \\ \varpi_{\mathrm{dl}} \end{bmatrix}. \end{aligned}$$

2.4 Finding A Numerical Solution of the Optimal Mobile Sensor Motion Scheduling Problem

2.4.1 A Brief Introduction to RIOTS

RIOTS stands for "recursive integration optimal trajectory solver." It is a Matlab toolbox designed to solve a very broad class of optimal control problems as defined in (2.7). When executing under Matlab script mode, the following configuration files need to be provided: `sys_l.m, sys_h.m, sys_g.m, sys_init.m, sys_acti.m`. They are the l_o, h, g_o functions in (2.7) and two initial conditions, respectively. Detailed instructions on how to prepare these files and many sample problems can be found in [98]. The most important function in this optimal control toolbox is `riots` explained in detail in [99, p.73].

```
[u,x,f,g,lambda2] = riots([x0,{fixed,{x0min,x0max}}],u0,t,Umin,Umax,
          params,[miter,{var,{fd}}],ialg,{[eps,epsneq,objrep,bigbnd]},
                                        {scaling},{disp},{lambda1}).
```

The parameters useful for understanding our numerical experiments here are as the follows:

- `x0`: initial values of $\tilde{\mathbf{x}}$.
- `fixed`: a vector to specify which entries in `x0` are fixed and which entries are not. Later in Section 2.5, results for two configurations are presented by changing `fixed` which are cases of "fixed initial states" and "unfixed initial states", respectively. For the first case, the robots' initial conditions, $\mathbf{x}_0$, are fixed. For the second case, $\chi_{\mathbf{dl}}$ is fixed so that the robots start from the optimal starting positions.
- `x0min`, `x0man`: bounds of the initial conditions.
- `u0`: initial values of the control functions τ.
- `t`: time.
- `Umin`, `Umax`: bounds for τ.

The definitions of other parameters are described in [99].

2.4.2 Using Matlab PDE Toolbox Together with RIOTS

The sensitivity function is generated before the function call of `riots` by Matlab PDE Toolbox. The procedure of solving the sensitivity function amounts to finding the solutions of the followings equations:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \nabla \cdot (\kappa \nabla u) + 20 \exp(-50(x_1 - t)^2), \\ \dfrac{\partial g_{(1)}}{\partial t} = \nabla \cdot \nabla u + \nabla \cdot (\kappa \nabla g_{(1)}), \\ \dfrac{\partial g_{(2)}}{\partial t} = \nabla \cdot (x \nabla u) + \nabla \cdot (\kappa \nabla g_{(2)}), \\ \dfrac{\partial g_{(3)}}{\partial t} = \nabla \cdot (y \nabla u) + \nabla \cdot (\kappa \nabla g_{(3)}), \end{cases}$$

where $\nabla = (\partial/\partial x, \partial/\partial y)$. Note that there are three g functions since there are 3 parameters c_1, c_2, c_3 in Section 2.3.2.

2.5 Illustrative Simulations

2.5.1 Differential Drive vs. Omni-Directional Drive

In [52], the robot model is a simple kinematic model:

$$\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\end{bmatrix} = r\omega(t),\ \begin{bmatrix}x(0)\\ y(0)\end{bmatrix} = \begin{bmatrix}x_0\\ y_0\end{bmatrix}, \tag{2.10}$$

where $\omega(t)$ is the angular speed vector, and r is the radii of the wheels. Obviously, (2.10) is an approximation. In this paper, we refer a robot that subjects to the kinematic in (2.10) a proximal "omni-directionally-driven robot" since the velocity can be set arbitrarily. When the robot is differentially driven, we are interested to see the difference in the optimal sensor motion scheduling. The following four cases are compared first:

- case 1: Omni-directionally-driven robots starting from a fixed initial state vector.
- case 2: Differentially-driven robots with a fixed given initial state vector. Moreover, we consider two subcases. Subcase(2a) has an initial yaw angle of 15° and subcase(2b) of -15°.
- case 3: Omni-directionally-driven robots without a fixed initial state vector. We assume that the optimal static sensor location problem is solved first. Use this obtained optimal position as the initial states and seek the optimal sensor motion trajectories.
- case 4: The same as in case 3 but using differentially-driven mobile robots.

According to the above definitions, Fig. 2.2 shows the results for case 1; Fig. 2.3 for case 2(a); Fig. 2.4 for case 2(b); Fig. 2.5 for case 3; and Fig. 2.6 for case 4. From these figures, we have the following observations:

- Differentially-driven robots are less likely to change the orientation. The optimal mobile sensor trajectories in cases 2 and 4 have smaller curvatures compared with that in cases 1 and 3.

- No matter what the robot dynamics are, the robots tend to move along the same trend. This can be observed by comparing cases 1, 2(a), 2(b) and cases 3, 4.

- For multi-robot cases, the final positions of the robots tend to be evenly distributed. Comparison on Fig. 2.3 and Fig. 2.4 is especially interesting. The two figures support each other, as the trend of the trajectories align along with each other.

2.5.2 Comparison of Robots with Different Capabilities

Here we consider two more cases to see .

- case 5: using a single "weak" robot, whose weight is 0.5 and the range of its torque for each wheel is ± 10.

- case 6: using a single "strong" robot, whose weight is 0.05 and the range of its torque for each wheel is ± 100.

With the same fixed initial states and the same time interval, the robot in case 5 moves shorter than in case 6, as seen from Fig. 2.7 and Fig. 2.8. This matches our intuition that it is desirable for the sensors to measure the DPS states at more spatial locations whenever possible.

2.5.3 On the Effect of the Initial Orientation

In additional to case 2(a) and case 2(b), the effects of different initial yaw angle is studies in this section. The robots associated with each figure in this subsection have the same mechanic configurations and the same initial conditions.

Let us compare the following figures:

- Figure 2.3: three robots with 15° initial yaw angle.

- Figure 2.4: three robots with -15° initial yaw angle.

- Figure 2.8: one robots with 15° initial yaw angle.

- Figure 2.9: one robots with -15° initial yaw angle.

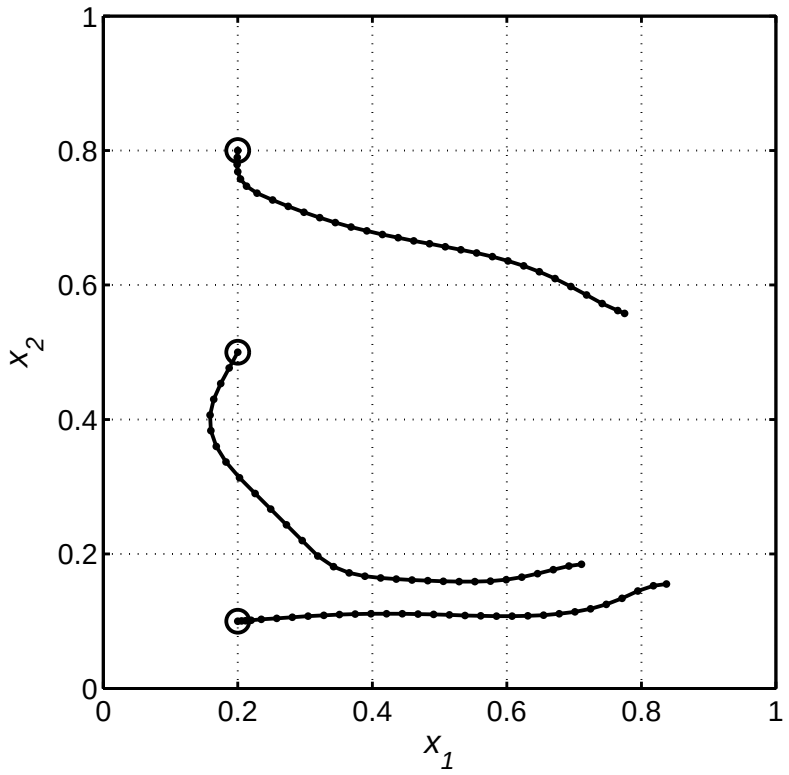


Fig. 2.2: The optimal sensor trajectories of omni-directionally-driven robots (case 1).

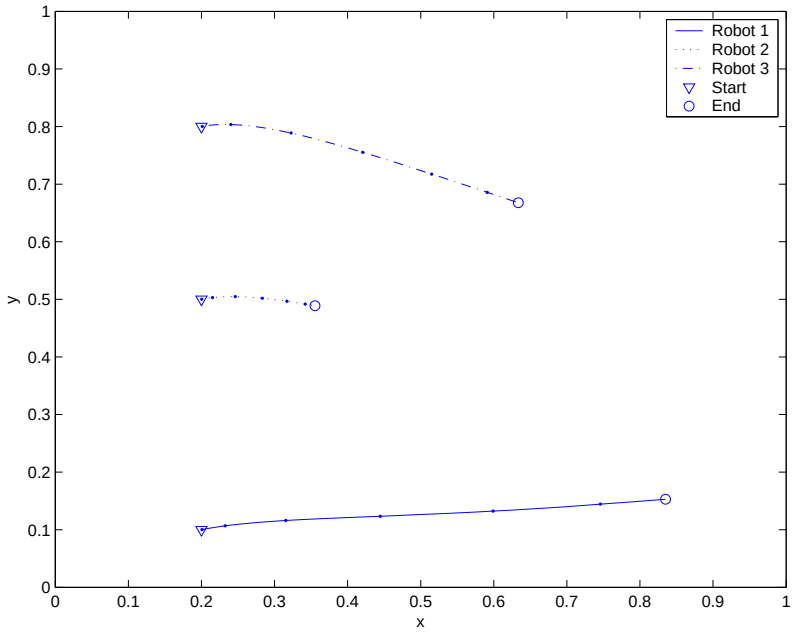


Fig. 2.3: The optimal sensor trajectories of differentially-driven robots: 15° initial yaw angle (case 2a).

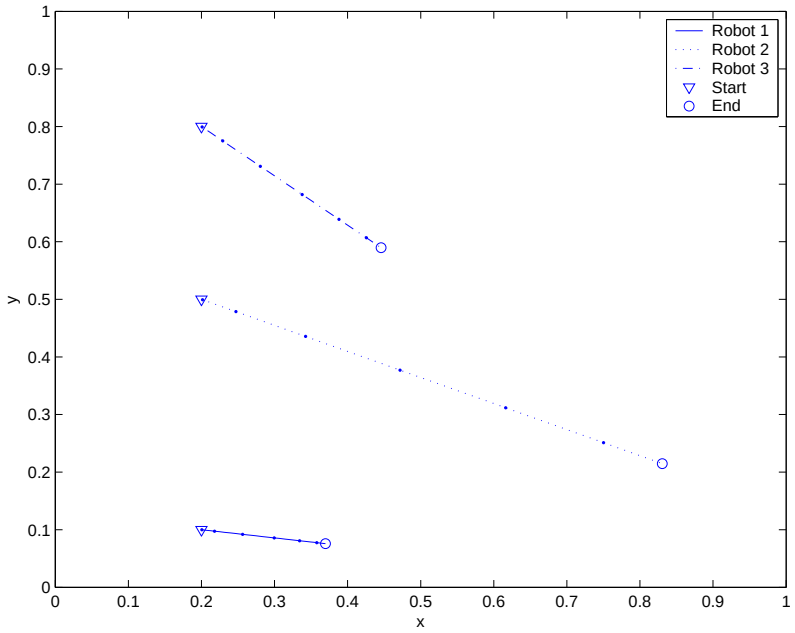


Fig. 2.4: The optimal sensor trajectories of differentially-driven robots: -15° initial yaw angle (case 2b).

The initial yaw angle affects the curvature of the optimal trajectory, but does not change the trend of the optimal trajectory. This indicates that the initial yaw angle matters, but not critical. Figures 2.8 and 2.9 support the above statement. With different initial yaw angles, the two robots starting at the same position have different trajectories, but their final positions are close. For multi-robot cases, the *formation* pattern of the robots' tends to be similar.

2.6 Chapter Summary

This chapter presents a numerical procedure for optimal sensor-motion scheduling of diffusion systems. Given a DPS with nominal parameters, differentially-driven mobile robots move along their optimal trajectories such that the sensor noise effect on the estimation of system parameters is minimized. This optimal measurement problem is an important module for a potential closed-loop DPS parameter identification algorithm. This chapter reformulates a differential-driven robot's dynamics model in the framework of optimal control. By the combined use of two existing Matlab toolboxes for optimal

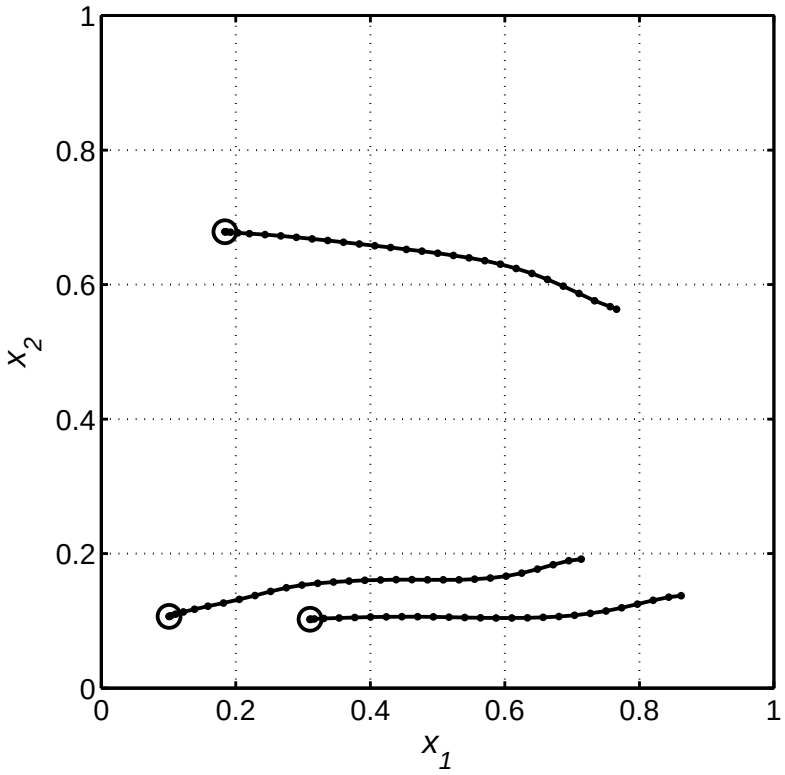


Fig. 2.5: The optimal sensor trajectories of omni-directionally-driven robots using optimal initial conditions (case 3).

control (RIOTS) and partial differential equations (Matlab PDE Toolbox), the optimal sensor-motion scheduling problem can be solved numerically. Simulation results and their observations are presented.

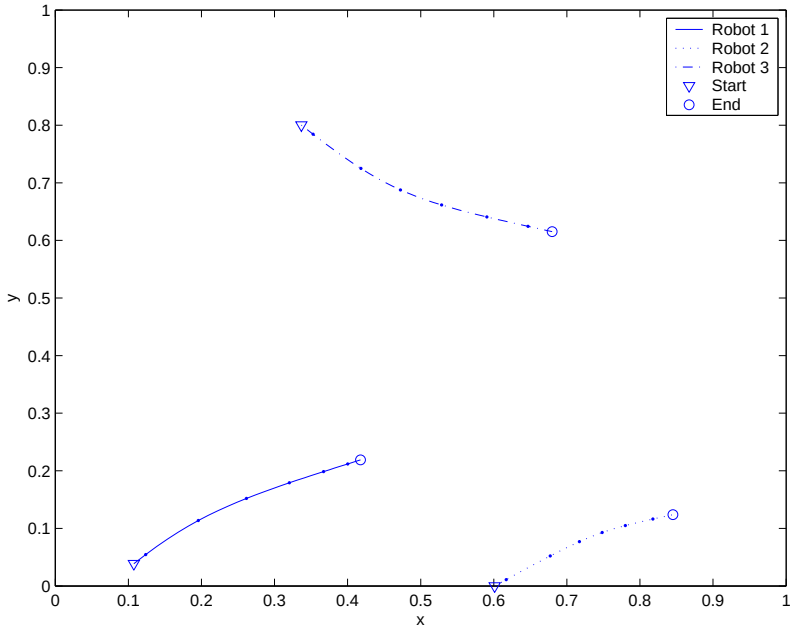


Fig. 2.6: The optimal sensor trajectories of differentially-driven robots using optimal initial conditions (case 4).

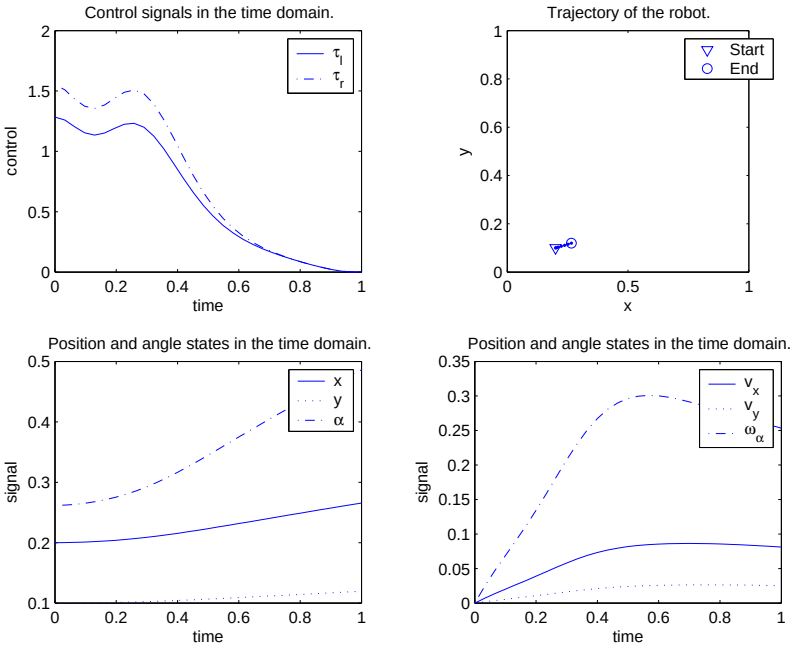


Fig. 2.7: The optimal trajectory of weak differential-drive robots.

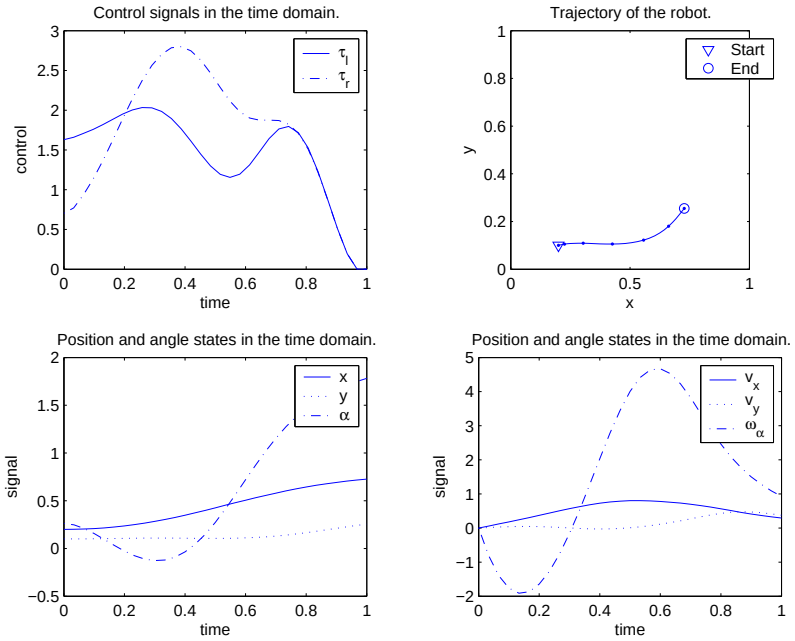


Fig. 2.8: The optimal trajectory of strong differential-drive robots: initial yaw angle is 15°.

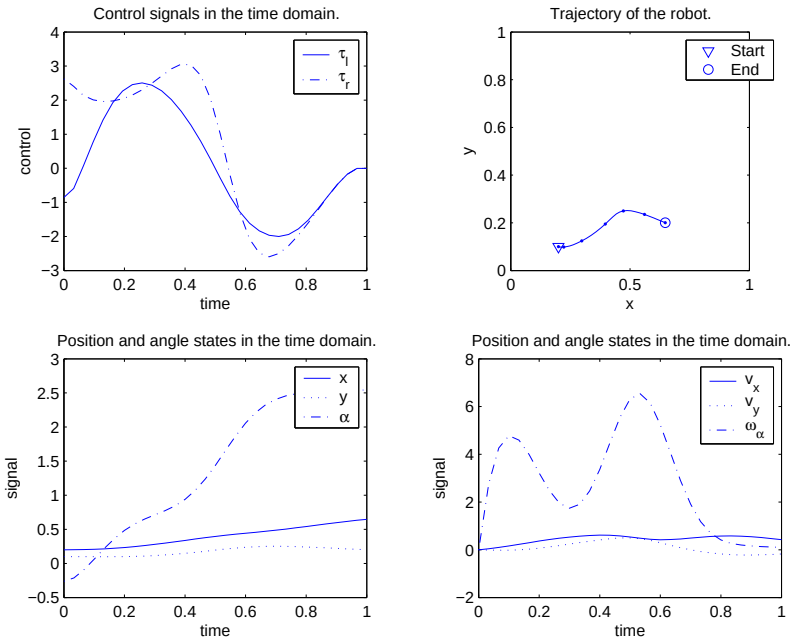


Fig. 2.9: The optimal trajectory of strong differential-drive robots: initial yaw angle -15°.

Chapter 3
Convex Optimal Sensor Selection (COSS): An Example of Implicit Optimal Sensor Selection Method

3.1 The Motivation and the Problem

Parameter estimation is an important application for WSNs, where physical parameters or quantities are under estimations. Sensor fusion plays an important role for parameter estimation. Many physical quantities of interest are either expensive or even impossible to measure by small sensors. For example, we may want to use several low-cost temperature sensors to estimate the position of a fire, instead of using an expensive IR camera. We may also want to use several gas sensors to locate the leaking source of certain invisible gas. The location can not be measured by a physical sensor directly. Some possible applications of parameter estimation include air pollution detection [25], wild fire monitoring [26], detection of persons and vehicles in open areas [27], and reconnaissance tasks [38].

A fundamental problem for those applications is to balance the trade off between the need to communicate useful information and the requirement to do so in an energy efficient manner. It is expected that a WSN, once deployed, will work for long periods of time with minimal human intervention. In most remote monitoring applications, individual sensor nodes run on a limited supply of energy (from batteries), and hence, excessive onboard processing and (perhaps more importantly) frequent invocation of the sensors radio, rapidly depletes the sensors energy source. On the other hand, if an energy conservation policy introduces too much estimation error, then the result is not acceptable. Thus, to meet these challenges, in recent years, a major research effort in modern computing has been devoted to the development of sensor network protocols and algorithms with energy-efficient capabilities.

3.2 Sensor Selection Problem and the Solution

3.2.1 Prior Art

Sensor selection (or sensor scheduling) is an important technique for energy efficient observation. In short, the sensor selection problem is defined as selecting the "proper" sensors and the "just enough" number of sensors to observe physical phenomena. The mathematical formulations on the "proper" and the "just enough" is not unique. However, the concept is similar: Balance the trade off between the need for precise estimation and energy cost for communication. For example, the "just enough" can be formulated as using the minimal number of sensors to send the data to the sink, subject to bounded estimation errors [48]. It is also possible to formulate the "just enough" as minimizing communication costs by sending only the non-redundant sensor information through the communication channel. The physical phenomena are estimated based on the less redundant data [49]. In this chapter, we argue that the minimum number of sensors that allowed by the Carathéodory's theorem is "just enough" to approximate the most precise estimate: The Cramér-Rao lower bound (CRLB).

Note that the brute force sensor selection approach is too expensive, in term of computation and hardware costs. Each sensor has two states: selected and unselected. Thus, the solution space for n sensors has 2^n states. This is a NP hard problem [100] and it is impossible to use the brute force approach in practice.

The sensor selection problem has been discussed from different aspects in the literature. Our classification on the existing sensor selection methods is shown in Fig. 3.1. Our proposed COSS method is an implicit optimization method. Hereby we briefly describe each method and compare them with our proposed method. For presentation purposes, we use a lamp tracking scenario has the example, even though our method is applicable to more general parameter estimation problems. In our scenario, sensor nodes equipped with light sensors are placed on a table. A lamp is placed at a constant height above those sensors and can be moved by hands. The task is to track the position of the lamp based

on the data collected by those light sensors. We assume that the locations of the sensors are known. [1]

On the highest level of Fig. 3.1, there are semantic and numerical methods. From the database aspect, the semantic method considers the sensor selection as a semantic routing problem [17, 54]. This method can select sensors based on properties of the sensor nodes, or simple properties of the sensor data. For example, the method is capable to select "all sensors on the 4th floor" or "all sensors who detect high temperature" [17]. This problem is essentially formulated as a tree search problem [54]. Our application scenario is different: There is no simple threshold to choose the proper sensors. For example, if several sensors are close to each other and have high light values, we may choose just one out of them since much of their sensor data are redundant. There is no simple threshold method to tell which sensor data is better.

Numerical methods select the proper sensor based on quantitative computations. Sensors can be selected based on the intrinsic properties of their measurements, or the other sensors' measurements, based on sensor fusion technologies. The model free methods do not require models of the physical world. For example, utility-based sensor selection methods [102, 103] select sensors by maximizing utility functions. Our method is model based, thus different.

There are also many model-based sensor selection methods, by which the physical models are used. Since many physical quantities are distributed and subject to physical laws, some sensor data can be inferred from other data. Given the model, it is not necessary to estimate physical phenomena with all the sensor data.

A model based geometric method is presented in paper [100]. Based on geometrical analysis on camera-like sensors, the paper concludes that the sensor selection problem can be solved in polynomial time. The paper also observed that "the estimates that obtained

[1] Although the sensor positions are required by the sensor selection algorithms, the sensor localization problem and the sensor selection problem are mostly independent. It is common for sensor selection papers to assume that the positions of the sensors are known [48, 49, 100, 101].

by four sensors are as good as the estimates obtained form all sensors." This observation is in consistent with the algebraic analysis in this chapter. Section 3.2.5 presents more details.

Sensor selection problem can be studied from the aspect of information compression: The sensor data with high uncertainty or redundancy should not be transmitted via the communication channel. The uncertainty is measured by entropy and the redundancy can be measured by correlations. For example, the paper [49] presents a sensor selection method based on the entropy filtering and Bayesian theorem. This is a grid-based method: The area of interest is segmented into many small cells, and the entropy of the target's location is computed based on the probability mass function. Since smaller entropy indicates less uncertainties of the information, paper [49] proposes an algorithm to select proper sensors by minimizing the entropy. Thus, only the "good" information with more certainties is sent to the sink. The probability mass function can be recursively updated according to the dynamics of the target. The updating method is based on the Bayesian theorem. More details of this method is presented in [5,38,104]. Based on the same framework, paper [105] proposes a faster heuristic entropy-based sensor selection method.

Comparing to the above methods, our proposed sensor selection is significantly different. 1) Our proposed method is not grid-based. 2) Our method minimizes estimation error, instead of uncertainties in sensor data. 3) Our method implicitly minimizes the number of selected sensors and explicitly minimizes the estimation error.

Another example of the information compression approach is present in [14], where sensors are directly deployed to measure the physical quantity of interest. The measurements are subject to a tempo-spatial correlation model. The variance of the estimation noise is minimized by selecting proper sensors. Our application scenario is different: Firstly, we are interested in parameter estimation problem, where the quantity of interest can not be directly measured by sensors. Secondly, we observed from experimental data that our sensor data should be considered uncorrelated. Figure 3.2 is a plot of a 11×11 correlation coefficients matrix for the light sensors on Tmote Sky [106] sensor nodes. In the experiment, we uniformly placed 11 sensors under a lamp at different distances. Figure 3.2 shows

that the magnitudes of the cross-correlations between two sensors at different distances are much smaller than that of the auto-correlations (the diagonal values in the figure). In fact, the mean of the cross-correlation coefficients of the experiment data is only 0.1181. Thus, our sensor data should not be considered correlated. Our sensor selection method does not depend on cross-correlations and we have verified our sensor selection method on our hardware testbed, which is based on Tmote Sky sensor nodes.

Sensor selection can be discussed within the framework of optimal estimation, where the estimation error is normally used as the cost function of the optimization. In previous works [48, 107, 108] and [101], the sensor selection problem is formulated as a constraint 0-1 integer programming problem, where the estimation error is minimized. The number of sensor is given and used as the constraints. In [48], the sensor selection is formulated as a 0-1 integer programming (or binary integer programming) problem and solved by a branch and bound (B&B) method. In [108], the sensor selection problem is formulated as a combinatorial optimization problem as well as a binary integer programming problem. The problem is solved by a GA (genetic algorithm) software. Those methods are closest to our proposed method, but the difference is still obvious: Our method does not explicitly minimize or choose the number of selected sensors.

Our method is an example of the implicit optimization. The feature of this class of methods is that the number of selected sensors is not explicitly shown in the cost function or constraints of the optimization. Instead, the number of selected sensors is minimized simultaneously during the time when estimation error is minimized. It is desirable and may be counter-intuitive. Using the minimum number of sensors, the least estimation error can be achieved! Selecting more sensors does not necessarily improve the estimation precision. Rather, more properly scheduled sensors can cover a larger area for observation. Before presenting the details in Section 3.2, we summarize the key features of our method: (1) Simultaneously achieve the optimal estimate with the least number of sensors, as well as the least energy costs. No compromise is required between the estimation precision and energy cost. (2) Simple and fast. (3) The robustness of the method has been verified by extensive hardware experiments.

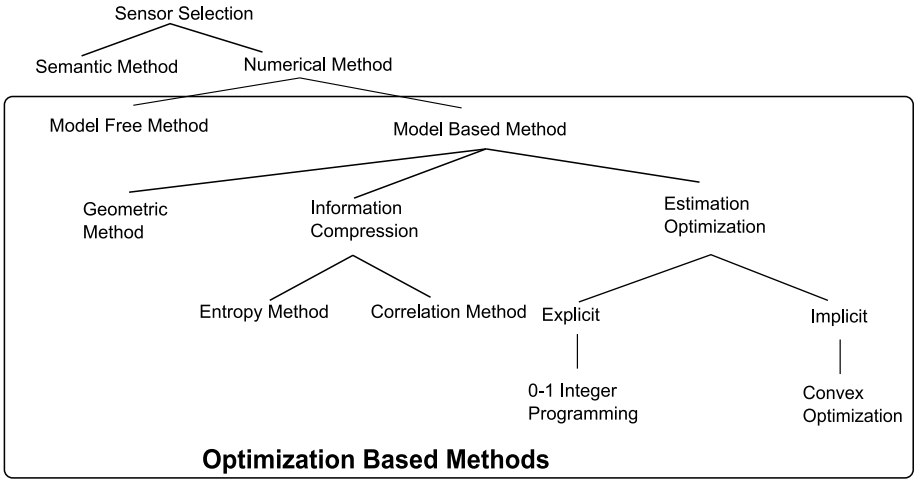


Fig. 3.1: A classification on sensor selection methods.

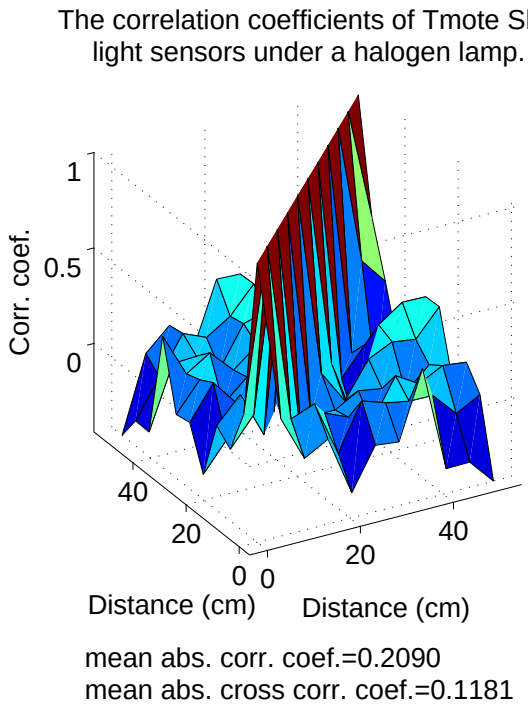


Fig. 3.2: The correlation coefficients of Tmote Sky light sensors. (Based on our hardware experiment data.)

3.2.2 Overview on Our Strategy

As aforementioned, the sensor selection problem is NP hard. Thus, our strategy is to simplify the problem by several reasonable approximations and solve the approximated problem by a fast optimization approach.

We do not add the two cost functions (cost for estimation errors and number of selected sensors) together and minimize the summation, or construct a vector based on the cost functions. These are standard multi-objective or combinatorial optimization techniques. Instead, we only minimize the cost function that associated with the estimation error. However, the number of selected sensor is automatically minimized during the time when we minimize the estimation error. This is due to the fact that we intentionally select a proper convex cost function and it has a convex constraint. According to the Carathéodory's theorem, the number of selected sensors is minimized. The details are presented in the Section 3.2.5.

We do not add the two cost functions (cost for estimation errors and number of selected sensors) together and minimize the summation, due to the high computation cost of integer programming. Remind that the number of selected sensors is an integer. Thanks to the gradient, the computation costs of continuous optimization problems are normally much less than the of the integer programming problems.

The system block diagram of our proposed method is shown in Fig. 3.3, where the physical parameter $\mathbf{q}^*$ is the position of lamp, y_i is the nominal light value, v_i is the sensor measurement noise, and s_i is the sensor reading. Note that our convex optimal sensor selection (COSS) algorithm includes the three blocks on the right, i.e., "wireless sensor network," "least squares fitting," and "optimal sensor selection." As we can see in Fig. 3.3 that the three blocks interact on each other, and they are inter-dependent. Thus, all the blocks are included in our COSS algorithm, which is list in Algorithm 3.1, Section 3.2.4. The "wireless sensor network" is associated to part 1 of the Algorithm 3.1 and carried out on individual sensor node. The part 2 in the Algorithm 3.1 is referring to the other two blocks, i.e., "least squares fitting," and "optimal sensor selection," in Fig. 3.3. The part 2 is the task for the sink.

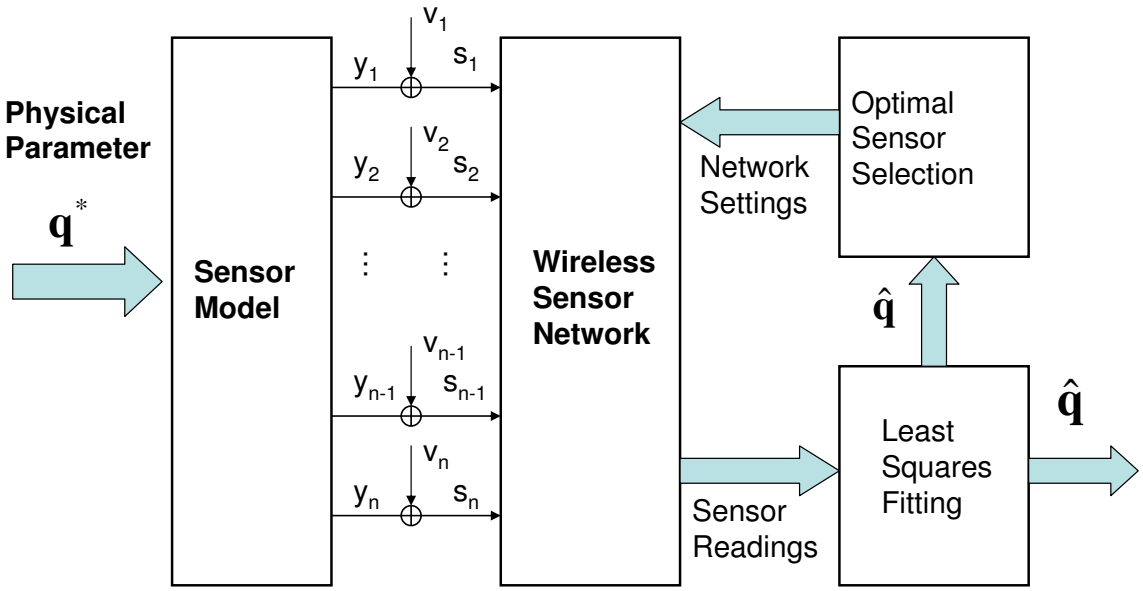


Fig. 3.3: System block diagram of the proposed method.

The future working scenario is shown in Fig. 3.4. In the figure, a target tracking scenario is demonstrated. However, the method is applicable to generic parameter estimation problems. After a target appears on the field of view, a subset of the sensor nodes observe the event and start measuring the physical quantity (light, temperature, gas concentration etc) that is associated with the target. At the beginning, many the sensor nodes detect the target and they enter an active mode. Their sampling rates and transmission rates are levitated to a high level, as shown in Fig. 3.4(A). Although the transmission rate is high, not too much energy is required, since this initial stage does not last for a long time. Soon, several sensors are selected to stay in the active mode and continuously monitor the target. Other sensors switch back to the inactive mode. Their sampling rates and transmission rates are considerably lower. One elected leader sensor among the active sensors takes the responsibility to report monitoring data to the sink. All these operations should be distributed, in future. This case is shown in Fig. 3.4(B). As the target moves, the selected sensors smoothly shift from one to another. This is shown in Fig. 3.4(C). Our current hardware implementation is presented in Section. 3.3. Although the current implementation is not at the level of this future working scenario, the experiment indicates

that the proposed COSS algorithm is fast and memory efficient. It has potentials to be implemented on low cost sensor nodes.

As aforementioned, the original sensor selection problem has a large solution space and it is hard to solve. Thus, our strategy is to simplify the problem by several reasonable approximations and solve the approximated problem by a fast optimization approach.

3.2.3 Problem Formulation

Assuming the true position of the lamp at the time instance k is $\mathbf{q}^*[k]$, and the position of the ith sensor is $\mathbf{r}_i$. The following equations hold:

$$\begin{aligned} y_i[k] &= f(\mathbf{q}^*[k]; \mathbf{r}_i), i = 1, 2, \cdots, n; \\ s_i[k] &= y_i[k] + v_i[k]. \end{aligned}$$

where f is the sensor model, $y_i[k]$ is the nominal reading from the ith sensor, and the associated real sensor reading is $s_i[k]$, which is corrupted by the independent Gaussian noise $v_i[k]$. In this chapter, we use a common energy model [51] as the sensor model.

Definition 3.2.1 (Energy Model)

$$\begin{aligned} y_i[k] &= \frac{c_1}{h^2 + d_i^2[k]}, \\ v_i[k] &\sim \mathcal{N}(0, \sigma), \\ d_i[k] &= \|\mathbf{r}_i - \mathbf{q}^*[k]\|, \end{aligned} \tag{3.1}$$

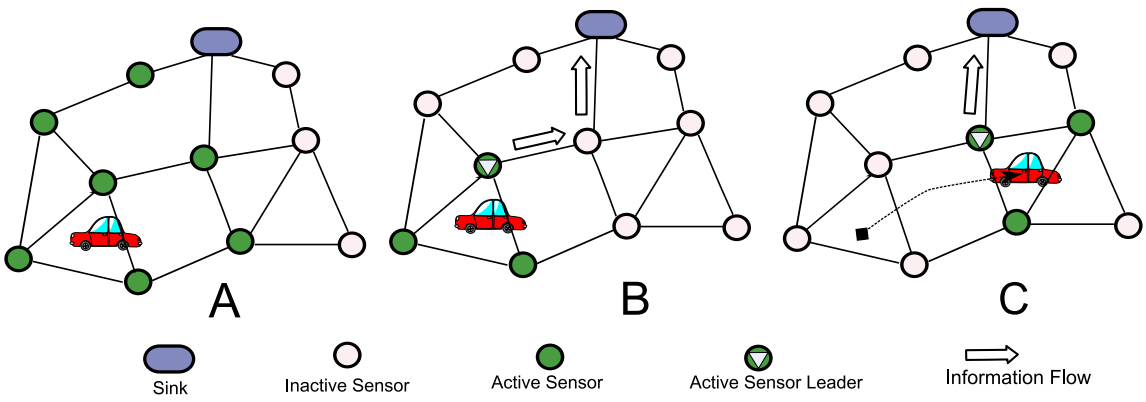


Fig. 3.4: The WSN sensor selection working scenario.

where h is the height of the lamp, $d_i[k]$ is the distance from the sensor to the exact position under the lamp, c_1 is a constant, and σ is the standard deviation.

Remark The energy model can be reasonably interpreted by physics. The fact that the sensor can observe the target indicates there is energy come from the target being detected by the sensor. The energy emitted from the target is uniformly spread in space and propagates in spheres. Thus, the energy density is proportional to $1/r^2$, where r is the radius of the sphere, or the distance from the target to the sensor. Under ideal conditions, if the sensor characteristics are linear, i.e., the sensor reading is proportional to the energy it received, then the energy model holds.

The configuration scenario of the energy model is shown in Fig. 3.5, where the target is on top of the plane that the sensor lay on. Easy to see that r^2 can be replaced by h^2+d^2. Thus, (3.1) is derived.

In the simulations of this chapter, we use the following constants

- $c_1 = 3.3032 \times 10^6$,
- $h = 20.32$ cm.

These parameters are chosen such that the energy model approximates the data measured by experiments.

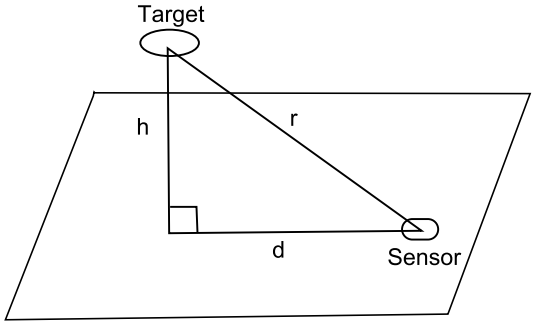


Fig. 3.5: Generic settings of energy model.

Definition 3.2.2 (Polynomial Model)

$$
\begin{aligned}
y_i[k] &= \begin{cases} c_2 + c_3 d_i[k] + c_4 d_i^2[k], & y_i[k] \geq y_L, \\ Invalid, & otherwise, \end{cases} \\
v_i[k] &\sim \mathcal{N}(0, \sigma_i[k]), \\
\sigma_i[k] &= \begin{cases} c_5 + c_6 d_i[k] + c_7 d_i^2[k], & y_i[k] \geq y_L, \\ Invalid, & otherwise, \end{cases} \\
d_i[k] &= \|\mathbf{r}_i - \mathbf{q}^*[k]\|,
\end{aligned}
$$

where c_2 to c_7 are constants, and $d_i[k]$ is the distance from the sensor to the exact position under the lamp.

Remark The polynomial model is characterized based on our hardware experiment data. The data are plot in Fig. 3.6. The left plot in the figure is the light value with respect to the distance, measured by cm. While the right plot is the standard deviation with respect to the distance. Note that the value of the standard deviation depends on the distance, in practice. In the figure, it is seen that the fitting errors of the polynomial model are relatively small.

The experiment data can be fitted by the energy model. However, the fitting error is larger than that of the polynomial model. This may due to the nonlinear characterizes of the sensors.

In the hardware implementations, we use the following parameters:

- $c_2 = 6720.0, \quad c_3 = -216.0, \quad c_4 = 2.524,$
- $y_L = 2500,$
- $c_5 = 824.0, \quad c_6 = -36.26, \quad c_7 = 0.4577.$

In the model, "invalid" means that the optimization for sensor selection does not consider the invalid sensors. Just as if those invalid sensors are not exist. The y_L is the threshold where the ambient light is more significant than the lamp's light. In Fig. 3.6, y_L is about 35 cm.

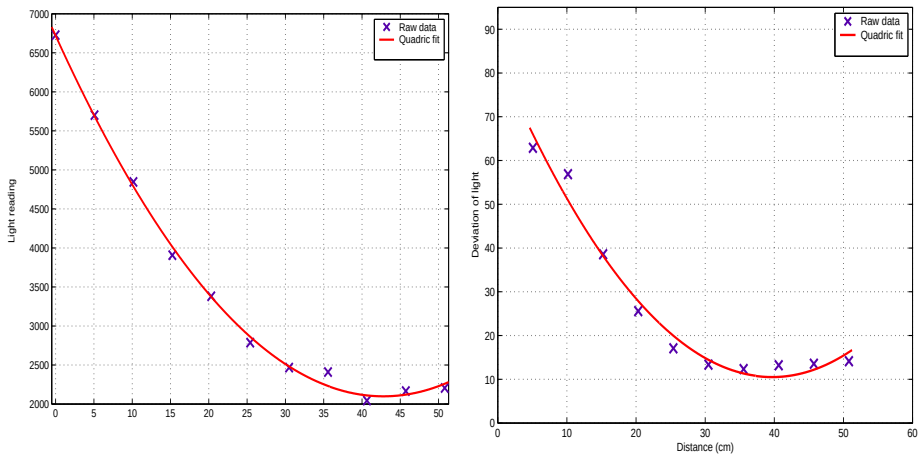


Fig. 3.6: The characteristics of the light sensors on Tmote Sky.

In order to reduce the noise v_i, the ith sensor node measures the n_i light values in the time slot t_S, averages them and sends the averaged value back to the sink. The averaged light value is $\bar{s}_i[k]$, whose standard deviation is smaller than that of the raw data:

$$\bar{\sigma}_i^2[k] = \sigma_i^2[k]/n_i.$$

If we can afford infinite number of samples, we can reject the noise totally. Of course, there is an upper limit on sample number in practice. For simplicity, we set n_S as the upper limit on the total number of samples for all the sensors in the time slot t_S. The intuitive interpretation is as follows. We estimate the position of an event based on sensor measurements. Since sensor measurements are noisy, the event observations are not perfect. Now, noise in the sensor measurements can be reduced by modifying the sensor design (use better hardware) or by filtering away sensor noises. Given fixed hardware, a simple filtering technique to adopt is to average the sensor data.

After receiving all the sensor data, the sink estimates the lamp's position by the standard nonlinear least squares (LS) method, and the output is $\hat{\mathbf{q}}_A$, which is also called the a priori position. For a network of n sensors, the $\hat{\mathbf{q}}_A$ is as (3.2).

$$\hat{\mathbf{q}}_A[k] = \text{argmin}_{\mathbf{q}} \frac{1}{2} \sum_{i=1}^{n} (\bar{s}_i[k] - y_i(\mathbf{q}; \mathbf{r}_i))^2. \tag{3.2}$$

Now, we introduce several approximations to simplify the problem.

- Instead of assigning each sensor a binary value that indicates the "selected" or "unselected" state, we assign a normalized sampling rate p_i to sensor i. That is,

$$p_i[k] \in [0,1] \text{ and } \sum_i p_i = 1.$$

  Thus, the integer programming problem is approximated by a continuous design problem.

- Our cost function is based on the Fisher information matrix (FIM), M, whose inverse matrix is the Cramér-Rao lower bound (CRLB) [52, 66]. The definition of Fisher information and FIM is presented in Appendix. B. Ideally, the optimal sampling rate is

$$\mathbf{p}^*[k] = \text{argmin}_{\mathbf{p}} \Psi(M(\mathbf{p}; \mathbf{q}^*[k])).$$

  Since $\mathbf{q}^*[k]$ is unknown, we replace it by $\hat{\mathbf{q}}_A[k]$, which is another approximation. Thus,

$$\hat{\mathbf{p}}[k] = \text{argmin}_{\mathbf{p}} \Psi(M(\mathbf{p}; \hat{\mathbf{q}}_A[k])).$$

- The nonlinear sensor models are linearized as $\mathbf{y}[k] = A^T \hat{\mathbf{q}}_A[k]$. This is a common approximation technique.

Thus, we simplified the original problem to the following sampling rate optimization problem.

Definition 3.2.3 (Sampling Rate Optimization Problem)

$$\begin{aligned}
\hat{\mathbf{p}}[k] &= \text{argmin}_{\mathbf{p}} \Psi(M(\mathbf{p}; \hat{\mathbf{q}}_A[k])), && (3.3)\\
\text{subject to : } \mathbf{p} &\geq 0,\\
\mathbf{1}^T \mathbf{p} &= 1,\\
\mathbf{y}[k] &= A^T \hat{\mathbf{q}}_A[k],\\
A &= \nabla_{\mathbf{q}} \mathbf{y}|_{\mathbf{q}=\hat{\mathbf{q}}_A[k]}, && (3.4)\\
M &= A \Sigma^{-1} A^T, && (3.5)\\
\Sigma^{-1} &= \begin{bmatrix} \tilde{\sigma_1}^{-2}[k] & 0 & \\ 0 & \tilde{\sigma_2}^{-2}[k] & \\ & & \ddots \end{bmatrix},
\end{aligned}$$

where $\tilde{\sigma_i}[k]$ is model dependent:

- For energy model: $\tilde{\sigma_i}^{-2}[k] = \sigma^{-2} p_i[k]$,
- For polynomial model: $\tilde{\sigma_i}^{-2}[k] = \sigma_i^{-2}[k] p_i[k]$.

The definition of $\Psi(M)$ follows (3.6).

$$\Psi(M) = -\ln\det(M). \tag{3.6}$$

A natural question to ask is why the Ψ function is defined as (3.6). This function is called D-optimality criterion in the literature of optimal experiment design (OED) [109]. In order to define the cost function as a scalar, the matrix M should be "scalarized" to a real number that indicates the "size" of the matrix. The D-optimality criterion is one but not the unique choice. Of course, if $\Psi(M)$ is also differentiable, then the optimization is easier.

Remember that $\det(M)$ is the volume of matrix M. Actually, when M is a 3 by 3 matrix, $\det(M)$ is the volume of the parallelepiped constructed by the three column vectors of M. In other words, $\det(M)$ is a metric to measure the size of M. Since $\det(M^{-1}) = 1/\det(M)$, $\det(M^{-1})$ is minimized when $-\ln\det(M)$ is minimized.

Commonly used optimality criteria include:

- The D-optimality criterion: $\Psi(M) = -\ln\det(M) = \ln\det(M^{-1})$.
- The E-optimality criterion: $\Psi(M) = -\lambda_{\max}(M^{-1})$.
- The A-optimality criterion: $\Psi(M) = -\mathrm{tr}(M^{-1})$.

Even though there are other optimality criteria available, D-optimality criteria is more commonly used. The unique feature of D-optimization compared to the rest is that the result of the D-optimization is not affected by linear transforms. That is, the result of the D-optimization is not affected by whether the unit of the measurements is in centimeters or in inches. In addition, the D-optimality criteria is differentiable. The computation on

D-optimization is easier than that for E-optimization, which is not differentiable. Although different optimality criteria may lead to significant different result, when M is not close to singular, the result of different optimality criteria are often close to each other. The details are presented in Section. 3.4.3

Now we prove that the FIM, M, is the inverse of the covariance of estimation error $\mathbf{e}$.

Theorem 3.2.4 *For the linear system* $\mathbf{s} = A^T\mathbf{q}^* + \mathbf{v}$*, where* v_i *is a zero mean noise with a standard deviation of* σ_i*, we have* M^{-1} *equals the covariance matrix of the estimation error. That is*

$$cov(\mathbf{e}) = M^{-1},$$

where $M = A\Sigma^{-1}A^T$, $\mathbf{e} = \hat{\mathbf{q}} - \mathbf{q}^*$ *and* $\hat{\mathbf{q}}$ *is an unbiased estimator of* $\mathbf{q}^*$. $\hat{\mathbf{q}}$ *is defined as* $\hat{\mathbf{q}} = \text{argmin}(A\mathbf{q} - \mathbf{s})^T\Sigma^{-1}(A\mathbf{q} - \mathbf{s})$.

Proof It is well known that $\hat{\mathbf{q}} = (A\Sigma^{-1}A^T)^{-1}A\Sigma^{-1}\mathbf{s}$. Because $\hat{\mathbf{q}}$ is unbiased, $E\{\hat{\mathbf{q}}\} = \mathbf{q}^*$.

$$\begin{aligned}
cov(\mathbf{e}) &= E\{(\mathbf{e} - E\{\mathbf{e}\})(\mathbf{e} - E\{\mathbf{e}\})^T\} \\
&= E\{(\hat{\mathbf{q}} - \mathbf{q}^* - 0)(\hat{\mathbf{q}} - \mathbf{q}^* - 0)^T\} \\
&= cov(\hat{\mathbf{q}}) \\
&= E\{(M^{-1}M\mathbf{q}^* + M^{-1}A\Sigma^{-1}\mathbf{v} - \mathbf{q}^*) \\
&\quad (M^{-1}M\mathbf{q}^* + M^{-1}A\Sigma^{-1}\mathbf{v} - \mathbf{q}^*)^T\} \\
&= M^{-1}A\Sigma^{-1}E\{\mathbf{v}\mathbf{v}^T\}\Sigma^{-T}A^TM^{-1} \\
&= M^{-1}.
\end{aligned}$$

In summary, $cov(\mathbf{e}) = M^{-1}$.

Since M^{-1} equals to the CRLB [52, 66], and $\Psi = \ln\det(M^{-1})$, the (3.3) minimizes the estimation error and pushes down the estimation error close to the CRLB.

Due to Theorem 3.2.15, which will be presented later, after the optimization, most sensors have sampling rates close to 0. No more than $m(m+1)/2$ sensors have higher sampling rates, where m is the number of parameters under estimation. Thus, we select

sensors whose sampling rates are higher than a threshold h_S. The set of selected sensor is $\mathbb{S}_S$.

$$\mathbb{S}_S[k] = \{i|\hat{p}_i[k] \geq h_S\}.$$

It is not difficult to tune the h_S in practice. Once (3.2.3) is finished, the sink turns off the sensors that have not been selected and collects the data from those selected sensors. Finally, the so called a posteriori position of lamp is estimated by the LS method.

$$\hat{\mathbf{q}}_B[k] = \text{argmin}_{\mathbf{q}} \frac{1}{2} \sum_{i \in \mathbb{S}_S[k]} (\bar{s}_i[k] - y_i(\mathbf{q}; \mathbf{r}_i))^2.$$

The system keep estimating the lamp by the selected sensors, until at certain time, $k+i$, when the a posterior position estimate, $\hat{\mathbf{q}}_B[k+i]$, has a big error, then we restart from the a priori estimation again. The error bound of $\hat{\mathbf{q}}_B[k+i]$ is estimated by its associated FIM, $M_B[k+i]$.

$$\begin{aligned} A_B &= \nabla_{\mathbf{q}} \mathbf{y}|_{\mathbf{q}=\hat{\mathbf{q}}_B[k]}, \\ M_B &= A_B \Sigma^{-1} A_B^T. \end{aligned} \tag{3.7}$$

In fact, if the target is smoothly moving, we can also restart from the sampling rate optimization. That is, just choose another 3 proper sensors to observe the lamp, instead of turning on all the 15 sensors and then select 3 sensors out of the 15. This strategy is more energy efficient. However, this approach has a limit on the dynamic of the target, i.e., the target can not move too fast. In the current scenario, the dynamic of target is assumed unknown. For the worst case, a target may shift from one side of the field of view to the other side in no time. The above strategy, although may cost more energy, is capable to capture the target's position under the worst cases: When the lamp is suddenly shifted from one side to the other side. For some applications, since the target is smoothly moving, it is desirable to consider the target dynamics for better estimation precision. This is our future work.

3.2.4 The Algorithm of the Convex Optimal Sensor Selection (COSS)

Definition 3.2.3 is formulated as an optimal experiment design problem, which can be solved by a multiplicative algorithm that updates $p_i[k+1]$ as $p_i[k]\phi_i(\mathbf{p}[k])/m$, where $\phi(\mathbf{p}[k]) = \nabla_{\mathbf{p}}\Psi$. It is proved that this method is a solution of the D-optimality criterion [52,110]. We use this method as a part of our convex optimal sensor selection (COSS) algorithm, which is listed as Algorithm 3.1. The computation of the algorithm is separated on the sensors and the sink.

3.2.5 Analysis on the Solution

Let us start from several definitions.

Definition 3.2.5 (Convex function) *A function $f : \mathbb{S} \mapsto \mathbb{R}$ is called convex over an open set $\mathbb{S}$ if*

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha)f(\mathbf{y}), \forall\mathbf{x}, \forall\mathbf{y}, \text{and } \mathbf{x}, \mathbf{y} \in \mathbb{S}_C, \alpha \in [0,1].$$

The function is called strictly convex if the $\leq$ sign is replaced by $<$.

Definition 3.2.6 *The solution domain is $\mathbb{S}_P$, which is a simplex defined as*

$$\mathbb{S}_P = \{\mathbf{p} | \mathbf{p} \in \mathbb{R}^n, \mathbf{1}^T\mathbf{p} = 1, \mathbf{p} \geq 0\}.$$

Definition 3.2.7 *Define the set of all possible FIM with the unified sampling rate as the set $\mathbb{S}_M$. That is,*

$$\mathbb{S}_M = \{M | \sum_{i=1}^{n} \mathbf{p}_{(i)}\mathbf{a}_i\mathbf{a}_i^T, \mathbf{a} \in \mathbb{R}^m, \mathbf{p} \in \mathbb{S}_P\}.$$

Next, we present an important result, which is the foundation of Theorem 3.2.15.

Lemma 3.2.8 *For the sampling rate optimization problem in Definition 3.2.3, if $Rank(A) = m$, the gradient is not zero inside $\mathbb{S}_p$. That is, $\nabla_{\mathbf{p}}\Psi(M) \neq \mathbf{0}$, $\mathbf{p} \in \mathbb{S}_P$.*

Table 3.1: Convex Optimal Sensor Selection (COSS) Algorithm

Part 1: On-sensor computation.

Receive t_S and n_i from the sink;
Collect n_i samples in the time slot t_S, and $\bar{s}_i$ is the average of those samples;
Wait for a small random time, then send $\bar{s}_i$ to sink;

Part 2: On-sink computation.

Initially $p_i = 1/n$ and `State`←`selection`;
if *State=selection* **then**
  Send n_i, t_S to the ith sensor;
  Wait for time t_S, collect $\bar{s}_i$;
  Estimate parameter $\mathbf{q}^*$ by the LS method, and the result is $\hat{\mathbf{q}}_A$;
  while *true* **do**
    if $\phi_i(\mathbf{p}[k+1]) < m + \eta, i = 1, 2, \cdots, N$ **then**
      exit the while loop;
    else
      $p_i[k+1] = p_i[k] \frac{\phi_i(\mathbf{p}[k])}{m}$;
    end
  end
  if $p_i[k+1] \geq h_S$ **then** $n_i = p_i[k+1] \times t_S$ **else** $n_i = 0$;
  Send $\mathbf{n}$ to the proper sensors and `State`←`tracking`;
end
if *State=tracking* **then**
  while *true* **do**
    Wait for time t_S, collect sensor reading $\bar{s}_i$, $i \in \mathbb{S}_S$;
    Estimate parameter $\mathbf{q}^*$ by the LS method. The result is $\hat{\mathbf{q}}_B$ and its associated FIM is M_B;
    if $\Psi(M_B)$ *is big* **then** `State`←`selection`, exit the while loop;
  end
end

Proof For simplicity, we ignore the time index $[k]$ in this proof. For simplicity, define $\mathbf{a}_i$ as $A^T = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n]$. Remind that $A^T \in \mathbb{R}^{m \times n}$, and A is defined in (3.4). So

$$\begin{aligned}
\Psi(M) &= -\ln\det(\sum_{i=1}^{n} p_i \mathbf{a}_i \mathbf{a}_i^T). \\
\frac{\partial \Psi(M)}{\partial p_j} &= -tr(M^{-1}\mathbf{a}_j \mathbf{a}_j^T), \\
\frac{\partial \Psi(M)}{\partial p_j} &= -\mathbf{a}_j^T M^{-1} \mathbf{a}_j, \\
\nabla_{\mathbf{p}} \Psi(M) &= \begin{pmatrix} -\mathbf{a}_1^T M^{-1}\mathbf{a}_1 \\ -\mathbf{a}_2^T M^{-1}\mathbf{a}_2 \\ \vdots \\ -\mathbf{a}_n^T M^{-1}\mathbf{a}_n \end{pmatrix}.
\end{aligned}$$

We now show that there exists some positive $\partial\Psi(M)/\partial p_j$.

Firstly, M is positive definite. To show this, we choose any $\mathbf{v} \in \mathbb{R}^m, \mathbf{v} \neq \mathbf{0}$, and multiply them to both sides of M

$$\mathbf{v}^T M \mathbf{v} = \sum_i p_i (\mathbf{v}^T \mathbf{a}_i)^2.$$

Since $Rank(A) = m$, or full rank, we have $Null(A^T) = \varnothing$. Thus, there does not exist a $\mathbf{v} \in \mathbb{R}^m, \mathbf{v} \neq \mathbf{0}$, such that $\mathbf{v}^T \mathbf{a}_i = 0$ for $\forall i \in [1, n]$. Or, $\mathbf{v}^T \mathbf{a}_i \neq 0$ for any $\mathbf{a}_i$.

In addition, since $\sum_i p_i = 1$, and $p_i \geq 0$, there is at least one positive p_i. That is, $p_k > 0$. Then

$$p_k (\mathbf{v}^T \mathbf{a}_k)^2 > 0.$$

For other $i \neq k$, we have

$$p_i (\mathbf{v}^T \mathbf{a}_i)^2 \geq 0.$$

So,

$$\sum_i p_i (\mathbf{v}^T \mathbf{a}_i)^2 > 0.$$

Thus, $\mathbf{v}^T M \mathbf{v} > 0$, i.e., M is positive definite.

Secondly, it is known that the inverse of a positive definite matrix is also positive definite. Since $Rank(A) = m$, there must exist $\mathbf{a}_i \neq \mathbf{0}$. Because M^{-1} is positive definite, for the none zero $\mathbf{a}_i$, i.e., $\mathbf{a}_i \in \mathbb{R}^m, \mathbf{a}_i \neq \mathbf{0}$, we have

$$\mathbf{a}_i^T M^{-1} \mathbf{a}_i > 0.$$

In summary, not all entries of $\nabla_{\mathbf{p}}\Psi(M)$ are equal to 0. Thus $\nabla_{\mathbf{p}}\Psi(M) \neq \mathbf{0}$.

Remark The geometric interpretation is shown in Fig. 3.7. In this 3D example, p_1, p_2 and p_3 are the axes. The parameter equation for the sliding surface is $p_1 + p_2 + p_3 = 1$. The simplex $\mathbb{S}_P$ is the triangle inside the (p_1, p_2), (p_2, p_3), and (p_1, p_3) plans. The color of the sliding surface indicates the value of a cost function: The value of the cost function increases as the color changes from blue to red. The triangle is a convex set. If there exists a global minimum point with the gradient equals to $\mathbf{0}$ (stationary point), Lemma 3.2.8 says that the global minimum point is out of the triangle, which is the same as Fig. 3.7. It is easy to see that the minimal value within the triangle, $\mathbb{S}_P$, must take place on the boundary of the triangle.

Actually, for our sensor selection problem, the stationary point ($\nabla_{\mathbf{p}}\Psi = 0$) is infinite far away, where $\mathbf{p} \to \infty$ and $M^{-1} \to 0$. In other words, the estimation error is totally rejected if we take infinite number of samples. It is easy to see that the minimum value on the triangle must take place on the boundary of the triangle. While the 3D example is straightforward, we need to prove that such relations are still true for arbitrary high order.

Next, we claim that $\mathbb{S}_M$ is convex for arbitrary high order. Since it is convex, the Carathéodory's theorem is valid on the set. Due to limited space, the proofs of several following theorems are ignored.

Lemma 3.2.9 *$\mathbb{S}_M$ is a convex set.*

Proof We want to prove the follows: if

$$M(\mathbf{p}_1) \in \mathbb{S}_M, M(\mathbf{p}_2) \in \mathbb{S}_M,$$

and

$$\bar{M} = \alpha M(\mathbf{p}_1) + (1 - \alpha) M(\mathbf{p}_2)$$

then $\bar{M} \in \mathbb{S}_M$, where $\alpha \in \mathbb{R}, \alpha \in [0, 1]$.

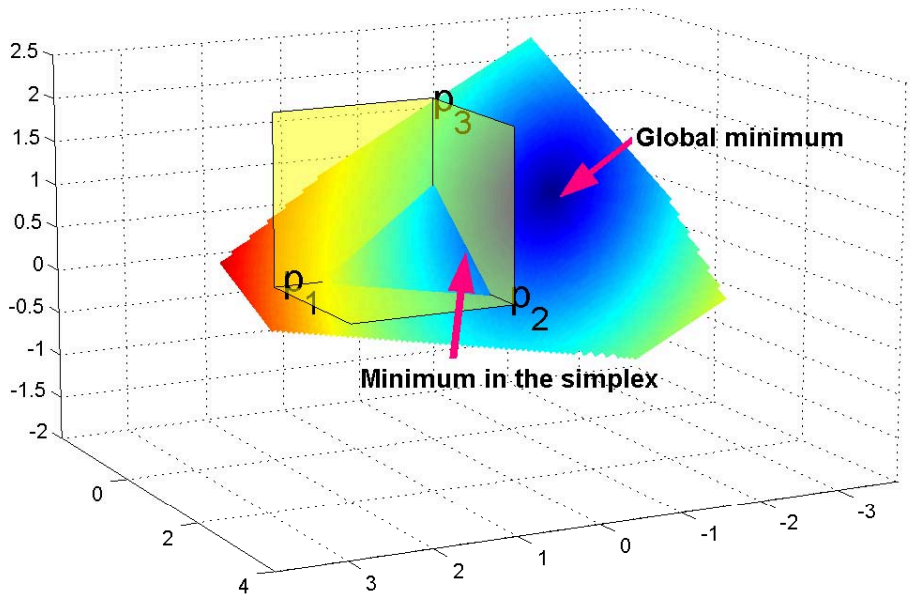


Fig. 3.7: A geometric interpretation on Lemma 3.2.8.

Define

$$\mathbf{p}_T := \alpha \mathbf{p}_1 + (1-\alpha)\mathbf{p}_2.$$

If

$$M(\mathbf{p}_1) \in \mathbb{S}_M \text{ and } M(\mathbf{p}_2) \in \mathbb{S}_M,$$

then we have

$$\bar{M} \quad = \quad \sum_{i=1}^{n} \mathbf{p}_{T(i)} \mathbf{a}_i \mathbf{a}_i^T,$$

Easy to see

$$\sum_{i=1}^{n} \mathbf{p}_{T(i)} = \alpha \sum_{i=1}^{n} \mathbf{p}_{1(i)} + (1-\alpha) \sum_{i=1}^{n} \mathbf{p}_{2(i)},$$

$$\sum_{i=1}^{n} \mathbf{p}_{T(i)} = \alpha + (1-\alpha) = 1.$$

So, $\bar{M} \in \mathbb{S}_M$.

Since $\mathbb{S}_M$ is convex, the Carathéodory's theorem is valid on the set.

Theorem 3.2.10 (Carathéodory's theorem, based on [111], p.72) *Let* $\mathbb{S}$ *be a subset of* $\mathbb{R}^n$*. Every element* $\mathbf{x}$ *in* $\mathbb{S}$ *can be expressed as a convex combination of no more than* $n+1$ *elements of* $\mathbb{S}$*. If* $\mathbf{x}$ *is on the boundary of* $\mathbb{S}$*,* $n+1$ *can be replaced by* n*.*

Remark An illustration on the Carathéodory's theorem in 2D domain is shown in Fig. 3.8. In the figure, points A to G are 2D points within the same convex hull. According to the Carathéodory's theorem, no more than 3 supporting points are required to express any point in the convex hull. It is intuitive to see that point F can be expressed as a convex combination of points A, B, and E, since F is inside the triangle ABE. This convex combination may not be unique. The point F can also be expressed by points A, C, and E. Note the words of "no more than." The point F is inside the convex hull and not on the boundary, but it can also be expressed as the convex combination of 2 points: i.e., points A and G.

Point B is on the boundary. According to the Carathéodory's theorem, it can be expressed by no more than two points. It is easy to see that point B can be expressed by points A and C.

We still need several theorems in order to prove that our COSS algorithm selects the minimum number of sensors that allowed by the Carathéodory's theorem.

Theorem 3.2.11 (Based on [110], p.81) *The D-optimality criterion is convex if* M *is non-negative definite and strictly convex if* M *is positive definite.*

Now, we extend the above theorem.

Corollary 3.2.12 *If* $\Psi(M)$ *is the D-optimality criterion, where* M *is as Definition 3.2.3, then* $\Psi(M)$ *is strictly convex.*

Proof Check the proof of our Theorem 3.2.8. Since M is positive definite, according to Theorem 3.2.11, the $\Psi(M)$ is strictly convex.

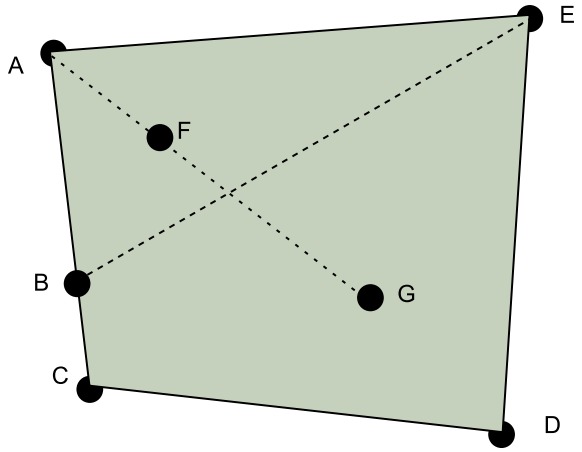


Fig. 3.8: An illustration of Carathéodory's theorem in 2D domain.

Remark As we are going to see in the proof of Theorem 3.2.15, the convexity of $\Psi(M)$ is important. A strict convex function has at most one unique minimum solution.

Theorem 3.2.13 (Based on [112], p.571) *If $\mathbb{S} \subset \mathbb{R}^n$ is a convex set and $g : \mathbb{S} \mapsto \mathbb{R}$ is a convex function, then a local minimum of g is also a global minimum. If in addition g is strictly convex, then there exists at most one global minimum of g.*

The multiplicative updating law in COSS can find the minimal solution, due to the following theorem.

Theorem 3.2.14 (Based on [110], p.140 and [52], p.65) *The multiplicative updating law ($p_i[j+1] = p_i[j]\frac{\phi_i(\mathbf{p}[j])}{m}$) in Algorithm 3.1 is nondecreasing. That is*

$$\lim_{j\to\infty} \det(M) = sup_{\mathbf{p}\in\mathbb{S}_P} \det(M),$$

where j is the iteration number.

Finally, we are ready to prove that our COSS algorithm selects the minimum number sensors that are allowed by the Carathéodory's theorem.

Theorem 3.2.15 *When $Rank(A) = m$, the COSS algorithm (Algorithm 3.1) selects no more than $m(m+1)/2$ sensors, where m is the number of parameters for estimation. That is, no more than $m(m+1)/2$ entries of $\hat{\mathbf{p}}$ are larger than 0, as the iteration number comes to infinity.*

Proof This theorem is an extension of Lemma 3.2.8, Lemma 3.2.9, Theorem 3.2.10, Corollary 3.2.12, Theorem 3.2.13, and Theorem 3.2.14: Theorem 3.2.14 says that the multiplicative updating law finds the minimal point after enough iterations. Lemma 3.2.8 says that there is no stationary point, where the gradient equals to 0, inside the simplex $\mathbb{S}_P$. So, Algorithm 3.1 must find a minimal point on the boundary of $\mathbb{S}_p$. The supporting points for M are M_i, where

$$M_i = \mathbf{a}_i \mathbf{a}_i^T,\ \mathbf{a}_i \in \mathbb{R}^m.$$

Each supporting point is a vertex of the simplex $\mathbb{S}_p$, since when

$$p_i = 1, p_j = 0 (i \neq j),$$

we have $M = M_i$; or, the convex hull of those supporting points is $\mathbb{S}_M$.

Note that M_i are symmetric matrices with $m(m+1)/2$ unique entries. Based on Lemma 3.2.9, $\mathbb{S}_M$ is a convex set and Theorem 3.2.10 is applicable. According to Theorem 3.2.10, there exits one solution to represent the optimal M by a convex combination of no more than $m(m+1)/2$ supporting points.

Corollary 3.2.12 says that the D-optimization function is strictly convex. According to Theorem 3.2.13, the one solution predicted by the Carathéodory's theorem is the unique solution.

In summary, Algorithm 3.1 finds the one and only one solution. Since the solution is on the boundary of the convex set $\mathbb{S}_P$, no more than $m(m+1)/2$ sampling rate p_i is not zero.

Remark In practice, infinite number of iterations are not possible. Thus, the practical solutions are always very close to the boundary but not perfectly on the boundary. According to Theorem 3.2.15, if the solution is perfectly on the boundary, no more than 3 sensors are required for a 2D tracking system, where $m = 2$. If the solution is close to the boundary but still within the simplex, no more than 4 sensors are selected for the same scenario. Remind that Theorem 3.2.10 predicts that one more supporting point is required if the element, i.e., the desired M, is inside the simplex.

Normally, in our tracking experiments, we observed that 3 sensors have high sampling rates (more than 0.2, or 20% of the total sampling rate), 1 sensor's sampling rate is in the middle, and the rest sampling rates are much smaller (less than 0.01). If 4 sensors are selected, we can tune our system to select no more than 3 sensors by increasing h_S or decreasing η. The tuning drives the solution closer to the boundary of the simplex, thus less sensors are selected. The tuning is not difficult in practice.

Remind that paper [100] is in consistent with our analysis (see Section. 1).

A further study on Theorem 3.2.15 indicates that the key feature of the COSS algorithm is not the D-optimality criterion, nor the multiplicative algorithm. As we see than in the proof of Theorem 3.2.15, the details of the multiplicative method is not used. The reason why the COSS can select sensors is due to that fact that $\Psi(M)$ is a convex function on a convex set.

The following theorem formally present our observation.

Theorem 3.2.16 *For the problem in Definition 3.2.3, if $\Psi(M)$ is convex, $Rank(A) = m$, and there exists an algorithm to find $\hat{\mathbf{p}}$, such that*

$$\hat{\mathbf{p}} = \text{argmin}_{\mathbf{p}} \Psi(M(\mathbf{p}; \hat{\mathbf{q}}_A[k])), \tag{3.8}$$

then no more than $m(m+1)/2$ entries of $\hat{\mathbf{p}}$ are positive. If (3.8) is not minimized, then no more than $m(m+1)/2+1$ entries are positive.

Proof Based on Theorem 3.2.15, the proof is trivial: Replace the "multiplicative algorithm" by the optimization algorithm which finds $\hat{\mathbf{p}}$, then this theorem is proved.

Remark This theorem reveals the existing of a class of implicit optimal sensor selection (IOSS) methods. Under the guidance of Theorem 3.2.16 more IOSS algorithms can be systematically designed. COSS is just a simple example of the many IOSS methods. It is our future work to design and compare more IOSS methods.

The theorem also gives us some guidance on parameter tuning of generic IOSS methods. Other IOSS methods can be tuned similar to that of the COSS. Please refer to the remarks of Theorem 3.2.15.

Remind that D-optimality criterion has similar properties to the A- and E-optimality criteria. In this chapter, we focus on D-optimality criterion. In addition to the advantages of the D-optimality that we list after Definition 3.2.3, the solution the the D-optimization, i.e., the multiplicative method, is simple and fast. It has potentials to be implemented on low-cost sensor nodes, where computation resources are very limited.

As an extension of Theorem 3.2.15, we give a theorem on the sensor deployment density requirements and achievable energy save factor.

Theorem 3.2.17 *If each point sensor detects targets in its vicinity with a radius of r, sensor deployment density is ρ, and the energy cost of turning on all the sensors is e_A, then the minimal sensor density required by the COSS algorithm is ρ_L, and its associated energy requirement is e_S.*

$$\begin{aligned} \rho_L &= \frac{m(m+1)}{\pi r^2}, \\ e_S &= \frac{m(m+1)}{\pi r^2 \rho} e_A, \end{aligned}$$

where m is the number of parameters under observation.

Proof When a target appears, all the sensors within its vicinity will detect it. The detection range is r. Based on Theorem 3.2.15, the COSS algorithm selects no more than $m(m+1)$ sensors. Thus, within the disc with a radius of r, we need to deploy at least $m(m+1)$ sensors. The area covered by the disc is πr^2. Easy to see that the minimal sensor density is ρ_L with $\rho_L = \frac{m(m+1)}{\pi r^2}$.

Assuming there are n sensors deployed in an area with the square s. Thus, the number of sensors being selected by the COSS algorithm is $\rho_L s$. Assuming the energy cost is uniformly spread among all the sensors, we have

$$e_S = (\rho_L \cdot s \cdot e_A)/n.$$

Thus $e_S = (m(m+1) \cdot e_A)/(\pi r^2 \rho)$.

Remark This theorem gives a necessary condition for the COSS algorithm, or other IOSS methods, to work properly. This theorem provides guidance on sensor deployment. That is, the sensor density must be no less than ρ_L. What is r in practice? r is the detection range of a sensor and should be found based on the data sheet of the sensor. Remind the polynomial model in Definition 3.2.2 has a constant named y_L. From the sensor characteristics plot, such as Fig. 3.6, it is easy to find the distance that associated with y_L. This distance is the maximum allowed r.

3.3 Experiment Results

3.3.1 Simulations

Simulation results are shown in Fig. 3.9 to Fig. 3.16. In those simulations, 15 light sensors are spread 20.32 cm apart from each other. The sampling period, t_S, is 1 sec, and the total sample number is 100. The signal noise ratio (SNR) is 8 dbm. The a priori position in the figure is $\mathbf{q}_A$. The a priori confidence ellipsoid is plot based on M_A, the FIM that associated to $\mathbf{q}_A$. The real target is believed within the ellipsoid. The a posterior position is $\mathbf{q}_B$, which is computed after the sensor selection. It is clear that the a posterior estimation is more precise in this case. The location error of the a priori estimation is 5.5883 cm, while the error of the a posterior is 3.7749 cm. From Fig. 3.10 we see that the algorithm converges fast. In this example, it only take 4 iterations for the sampling rates to converge close enough to the optimal value.

Note that the COSS algorithm does not requires the sensors to be placed uniformly. 60 sensors are randomly placed for the example in Fig. 3.13. In this example, the positing error of the a priori estimate is 1.7038 cm and that of the a posteriori estimate is 1.6193 cm. The determinate of the FIMs that associated with the a priori and a posteriori estimates are 12.9581 and 10.9482, respectively.

The simulations indicate that the selected sensors are not necessarily the closest sensors to the target. In Figs. 3.11, 3.13, and 3.14, the closest one sensor is actually not selected. Figure 3.12 is the light field for the example in Fig. 3.11. We see that selected sensors have relatively high, but not the highest, light values.

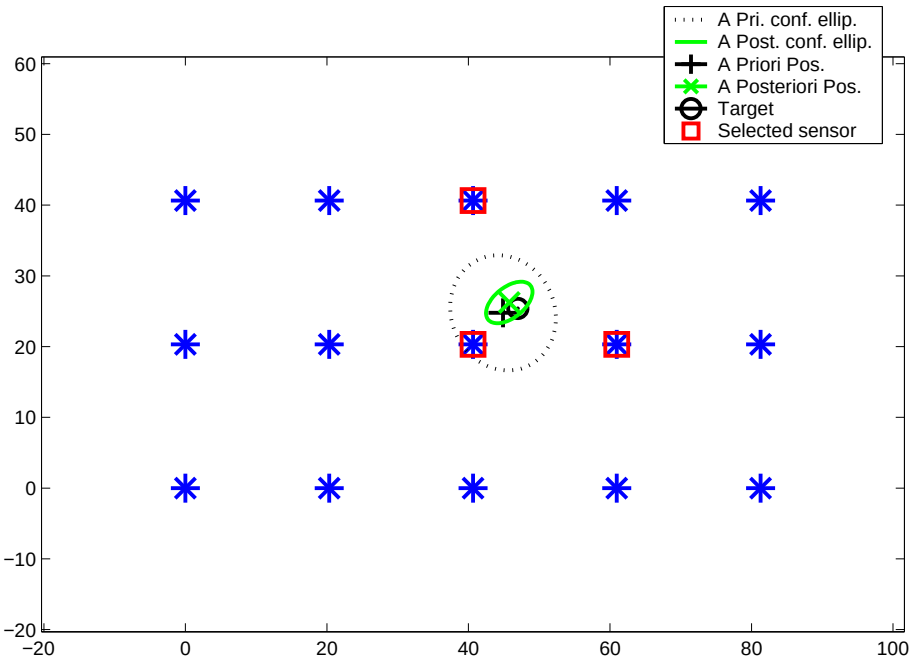


Fig. 3.9: A simulation based on the COSS algorithm.

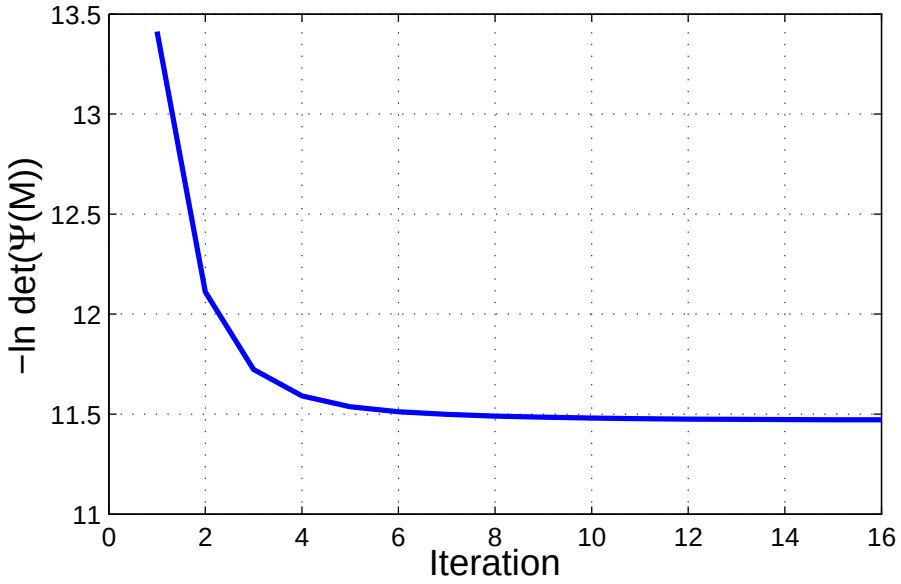


Fig. 3.10: Convergence speed of the COSS algorithm.

Figure 3.15 and Fig. 3.16 are some extreme cases to show the performance of the COSS algorithm. In the figures, the target is not inside the WSN. We see that when the target is out of the coverage of the WSN, but not too far away, the position estimate is still precise. When the target is too far away, both the a priori estimate measured by the LS method and the a posteriori estimate based on the sensor selection optimization can not locate the target properly.

3.3.2 Hardware Experiments

Since we introduced several approximations in the our algorithm, it is important to verify the validity of those approximations by our physical testbed. A picture of the testbed in shown in Fig. 3.17. In the testbed, 15 Tmote sky sensor nodes are placed under a halogen lamp. The sink Tmote sky node (not in the picture) is connected to a PC. A GUI is running on the PC.

Figures 3.18, 3.19, 3.20, 3.21 and 3.22 are captured frames from our movie [113] that demonstrates the testbed. Notation on Figures 3.18, 3.19, and 3.20 are manually added. Figure 3.18 shows the GUI running on the sink. Figure 3.19 shows the initial stage of the tracking. The track is clearly shown in Fig. 3.20. The important frames of the movie are shown in Figs. 3.21, 3.22. The picture at the left bottom is from a movie taken by a camcorder. The selected sensors are the green dots on the screen, the unselected sensors are the red dots. In the movie, we see that no matter the lamp is smoothly moving or suddenly shifting, the system can always track the lamp.

Figures 3.21 and 3.22 are frames from our movie [113] that demonstrates the testbed. The picture at the left bottom is from a video taken by a camcorder. The selected sensors are the green dots on the screen, the not-selected sensors are the red dots. No matter the lamp is smoothly moving or suddenly shifting, the system can always track the lamp.

Comparing to the simulation, the hardware implementation is more challenging due to the following reasons:

- Imperfect communications introduce packet drops.

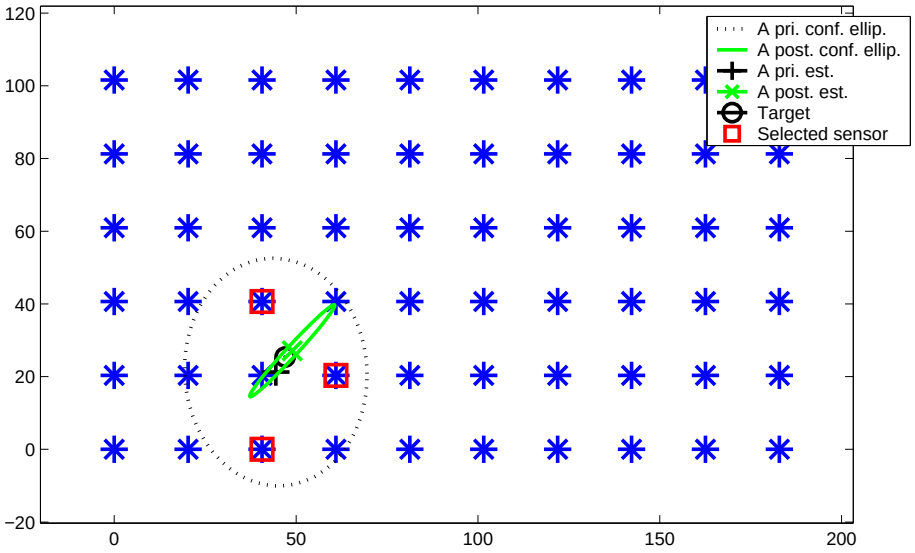


Fig. 3.11: Sensor selection for a network of 60 sensors by COSS.

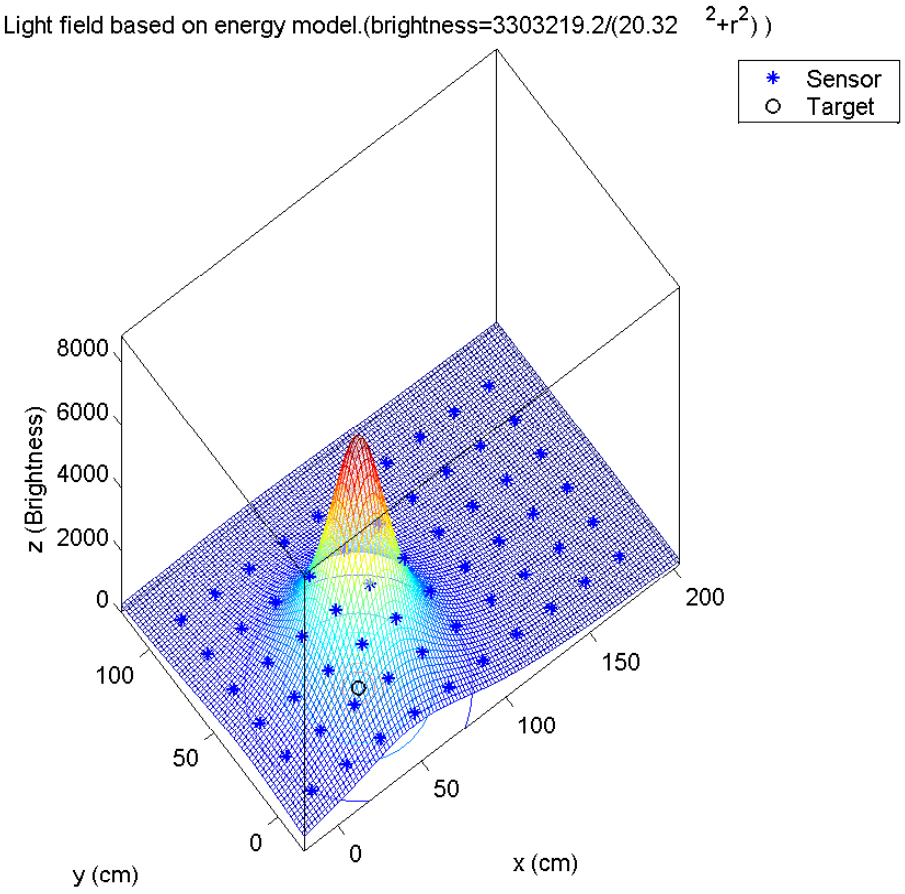


Fig. 3.12: Light field.

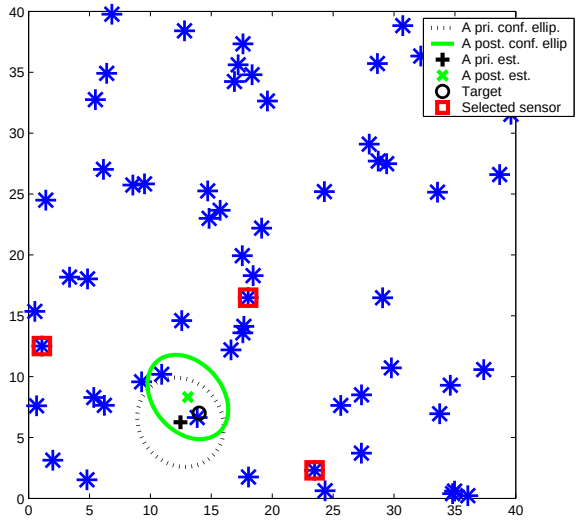


Fig. 3.13: An example of applying the COSS algorithm to randomly placed sensors.

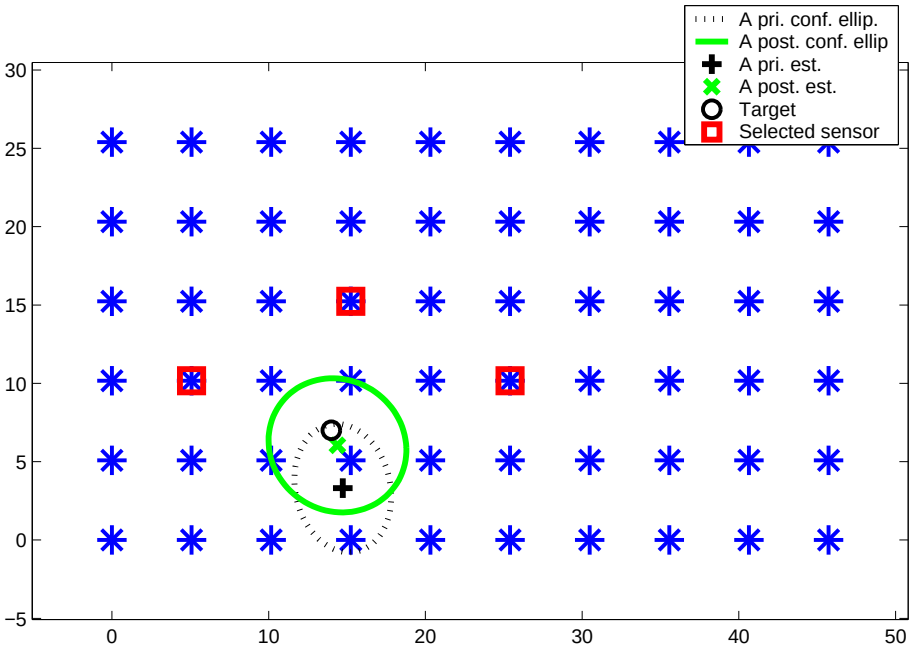


Fig. 3.14: Applying the COSS algorithm to densely deployed sensors.

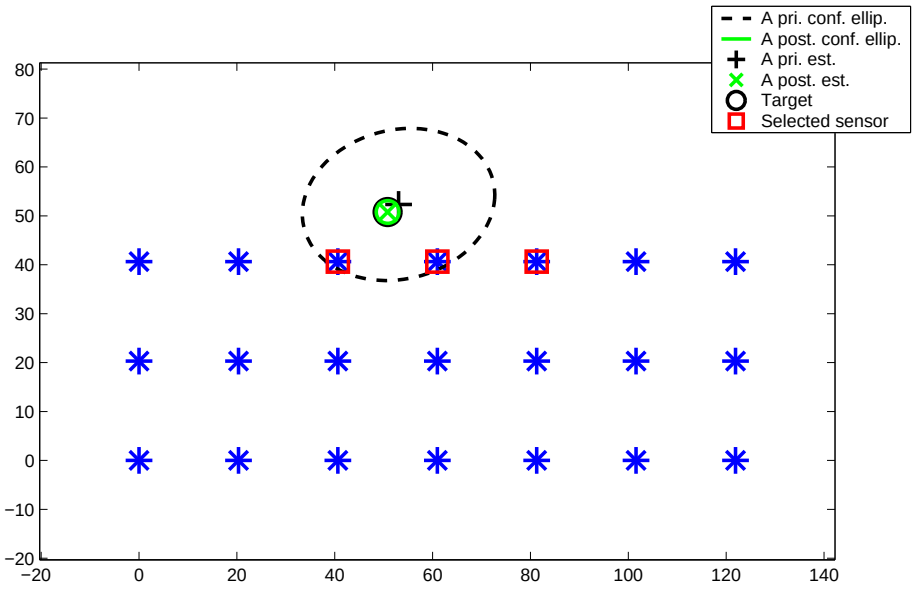


Fig. 3.15: Target is out of the boundary of the WSN.

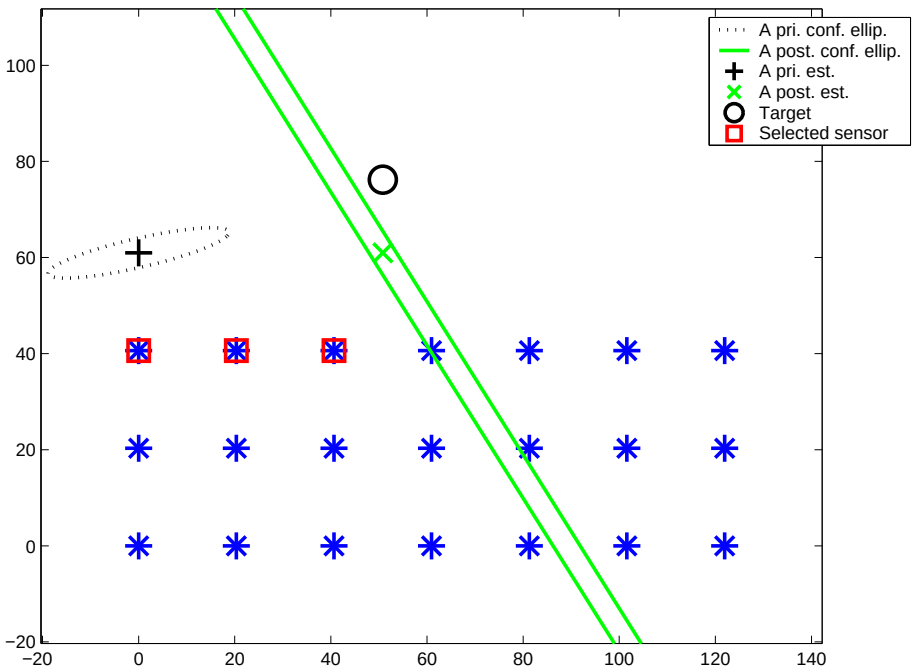


Fig. 3.16: Target is far out of the boundary of the WSN.

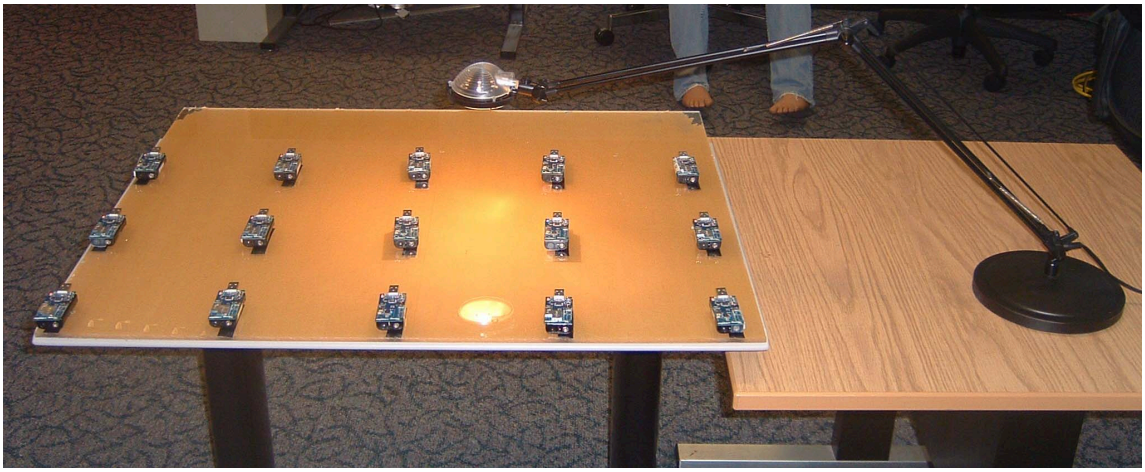


Fig. 3.17: A picture of our sensor selection testbed.

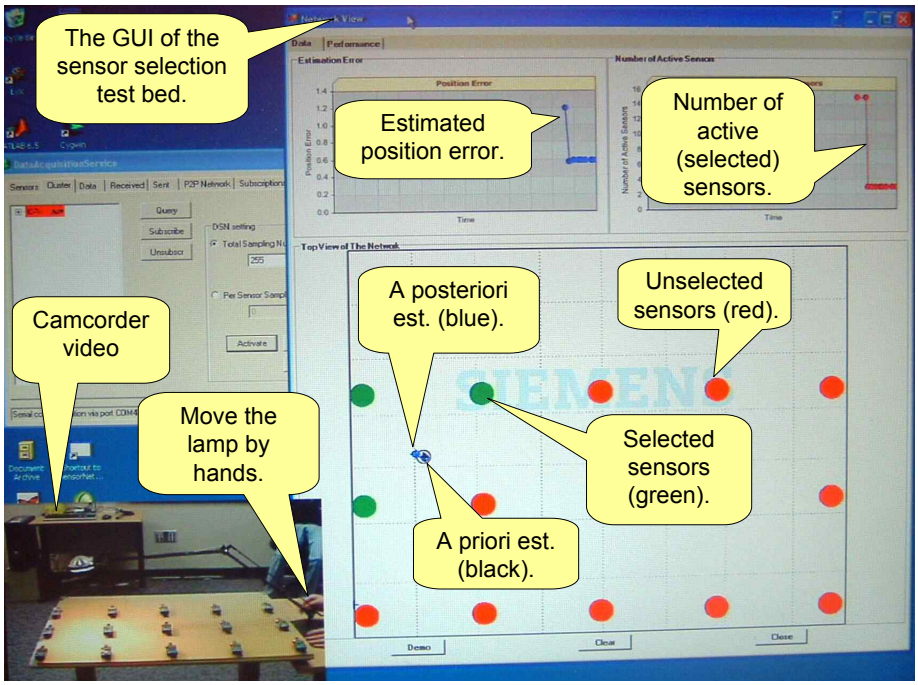


Fig. 3.18: A screen shoot of our sensor selection testbed: The GUI.

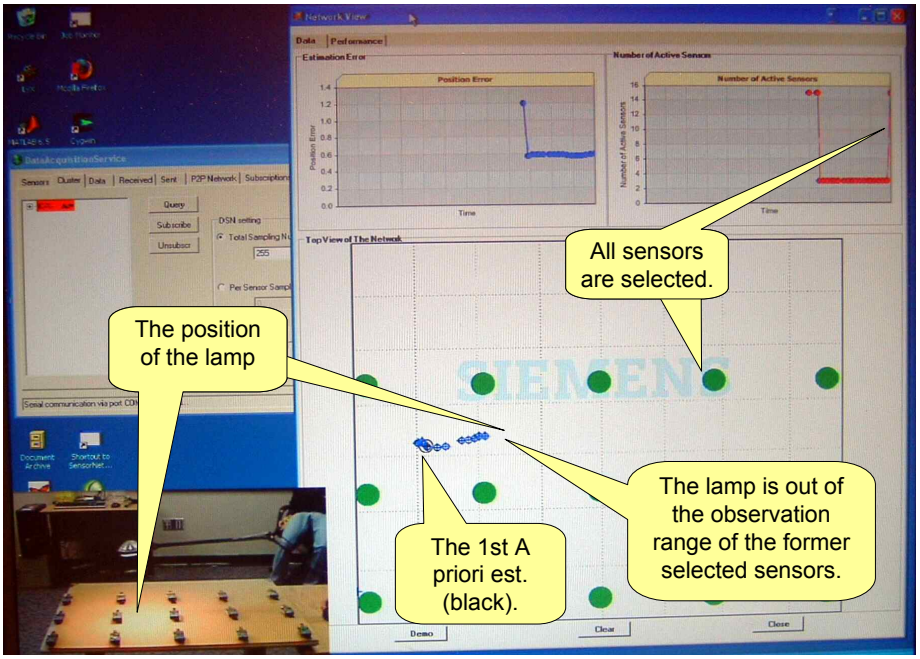


Fig. 3.19: A screen shoot of our sensor selection testbed: Tracking the lamp.

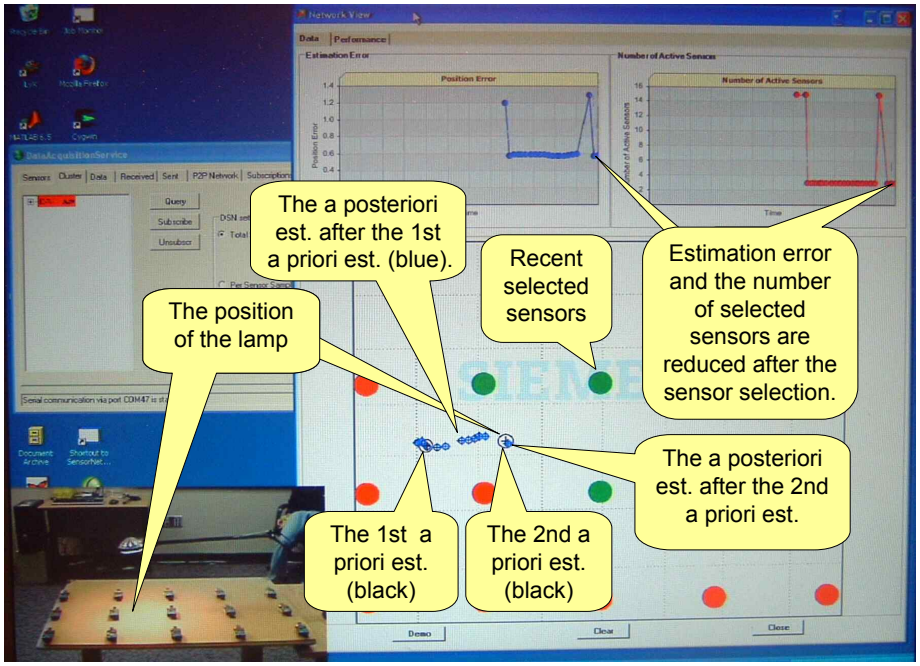
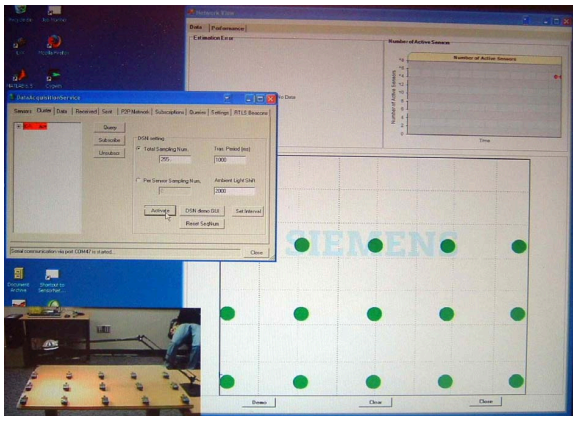
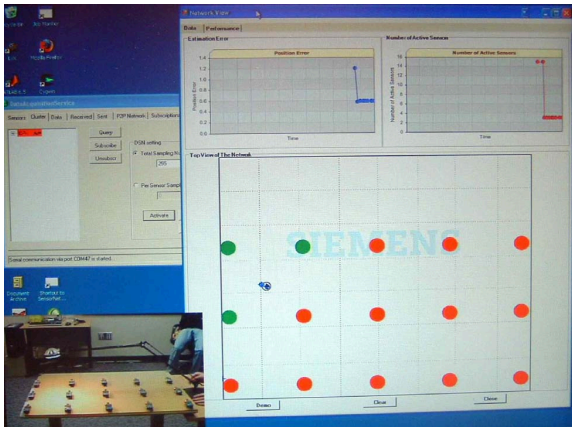


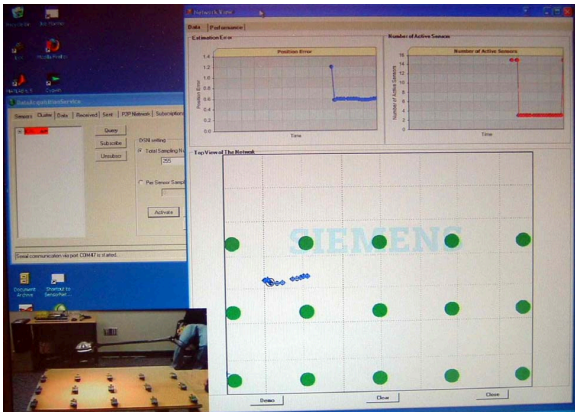
Fig. 3.20: A screen shoot of our sensor selection testbed: The track.



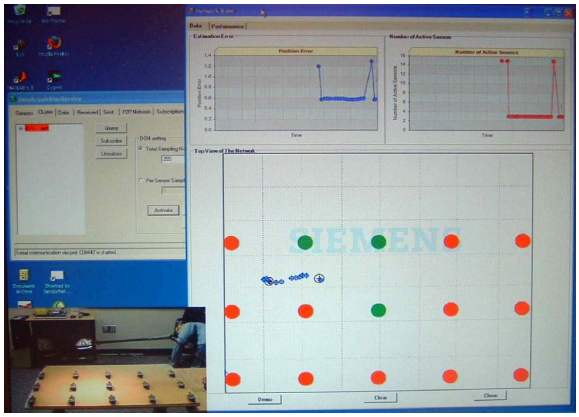
(a) Frame 1



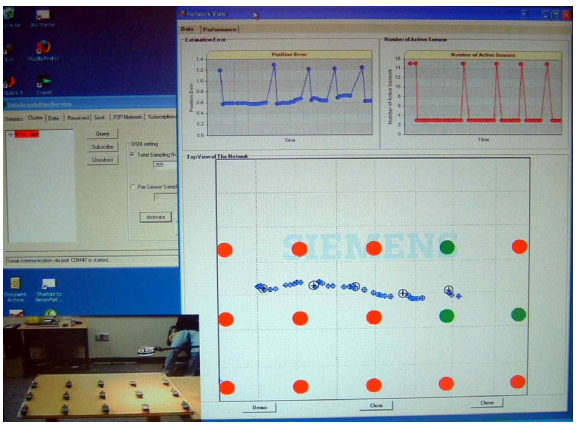
(b) Frame 2



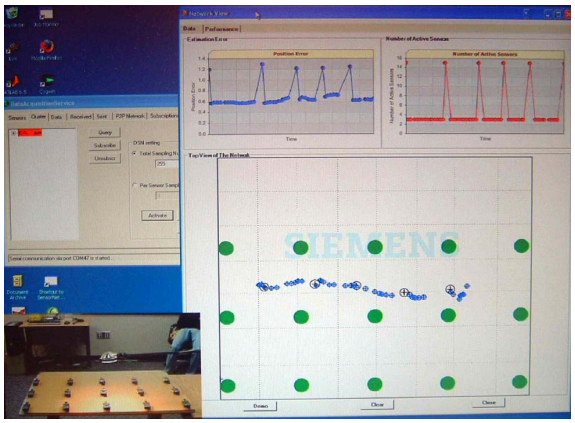
(c) Frame 3



(d) Frame 4

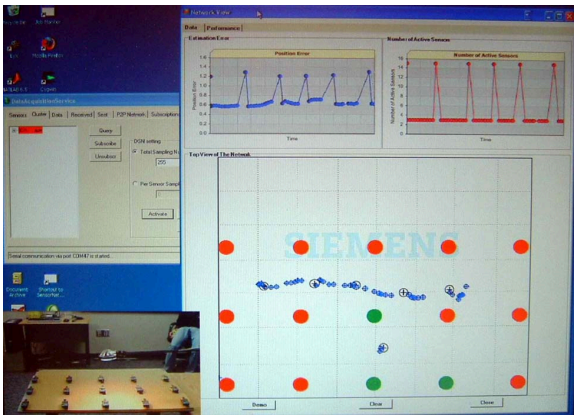


(e) Frame 5



(f) Frame 6

Fig. 3.21: Screen shots for tracking mobile target using the sensor selection algorithm.



(a) Frame 7

Fig. 3.22: Frame 7 for tracking mobile target using the sensor selection algorithm.

- Disturbances from the ambient light.
- The light bulb is not a perfect point light source. In fact, the size of the light bulb is about 3 cm, which is comparable to the tracking error of our testbed.
- The light reflector behind the light bulb is not perfect and the reflection is not very smooth. That is, the distribution on the light field is not perfect.
- The sensor noise is not an ideal Gaussian noise.

According to our experiment results, those approximations are valid under realistic application conditions. The algorithm is robust enough for us to demonstrate our system under different ambient light conditions. The system is also robust to other WiFi (IEEE 802.15.x) devices in our lab. No performance degradation is observed after introducing those WiFi devices in the vicinity of the testbed.

Despite of those difficulties, we still achieve our goals: The COSS algorithm selects 3 sensors and, statistically, the a posteriori estimates are more precise than the a priori estimates. We measure the positioning errors on 21 points on the testbed. The indices of those 21 points are shown in Fig. 3.23. After placing the lamp on one of those positions, we manually measure the position and take it as the real position. The a priori and a posteriori

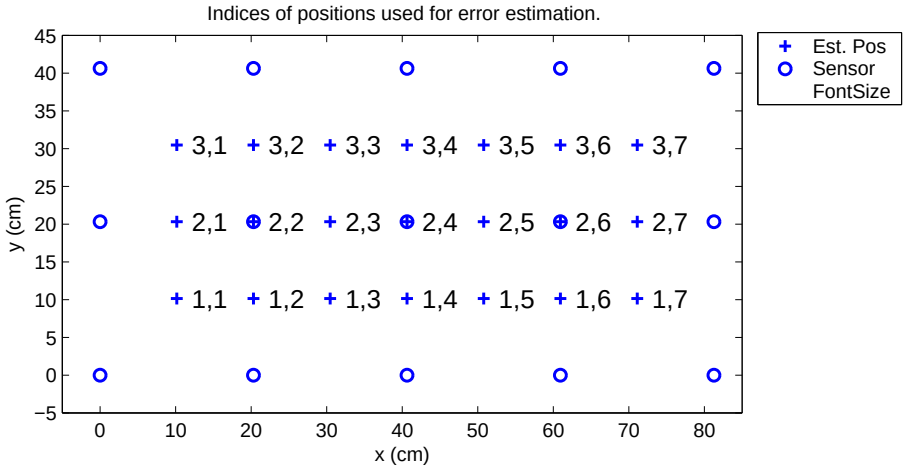


Fig. 3.23: Indices of positions used for error estimation.

positions are logged in a data file. Figure 3.24 is plotted based on those experiment data. The mean tracking error for the a priori estimate is 3.1387 cm, and the mean error for the a posteriori estimate is 3.0269 cm. Not surprisingly, the advantages of the a posteriori estimates over the a priori estimates in the hardware experiments are not as significant as that in the simulations. This is due to the listed hardware imperfectness. The upper two plots in Fig. 3.24 are drawn in scale. We see that the positing error, or the height in the z-axis, is very small. In order to show see the positing error clearly, the two lower plots in Fig. 3.24 have enlarged z-axis.

On Fig. 3.25 the ratio of the a posteriori estimation error over the a priori estimation error is plot as a surface. Thus, the COSS improves estimation precision at those places where that ratio is lower than 1. In the figure, we see that on most positions the error of the a posteriori estimate is smaller than that of the a priori estimate. Although the error of the a posteriori estimate is not always smaller than that of the a priori estimate, the a posteriori estimates are better for most of the area.

Figures 3.26, 3.27, 3.28, and 3.29 are captured from a movie [113] which is rendered based on the same experiment data that Fig. 3.24 and Fig. 3.25 use. The plots in Figs. 3.26, 3.27, 3.28 , 3.29 have information that have not been presented in Fig. 3.24 or Fig. 3.25. The manually measured positions of the lamp, the selected sensors, the a priori and a posteriori position estimates.

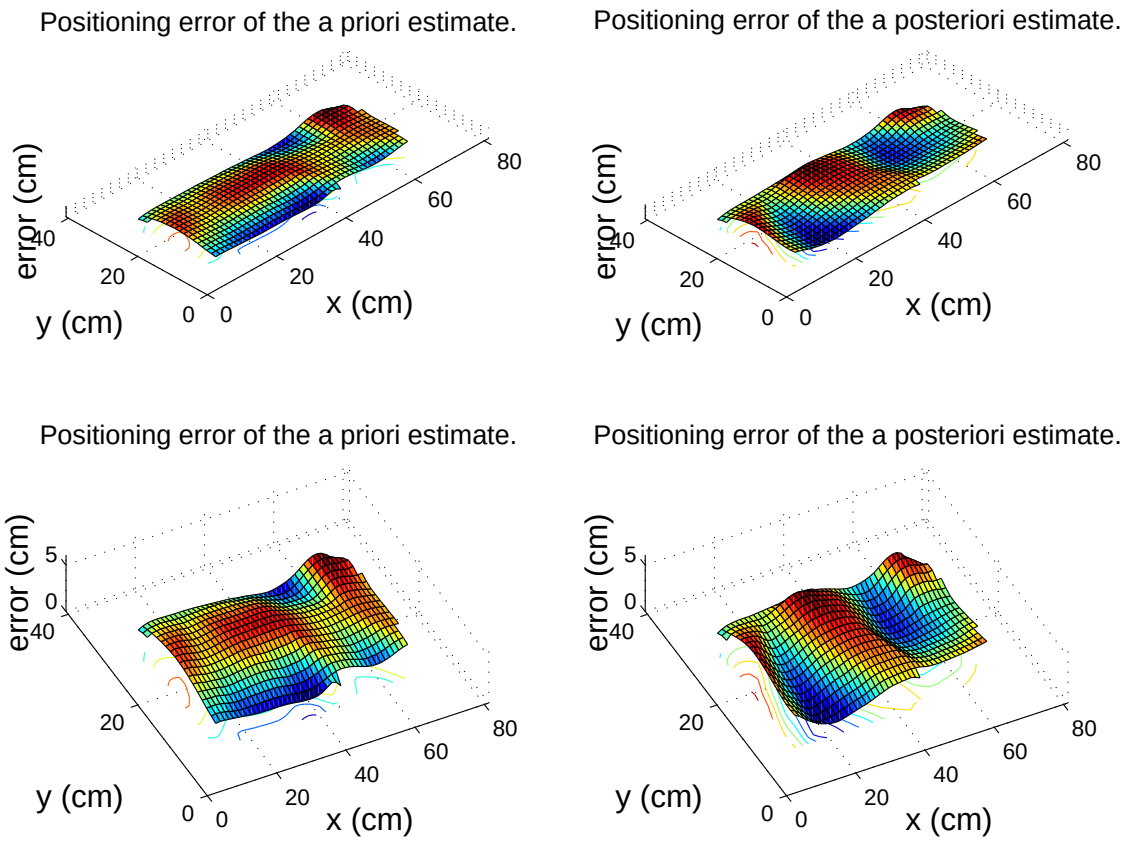


Fig. 3.24: Estimation errors of our sensor selection testbed.

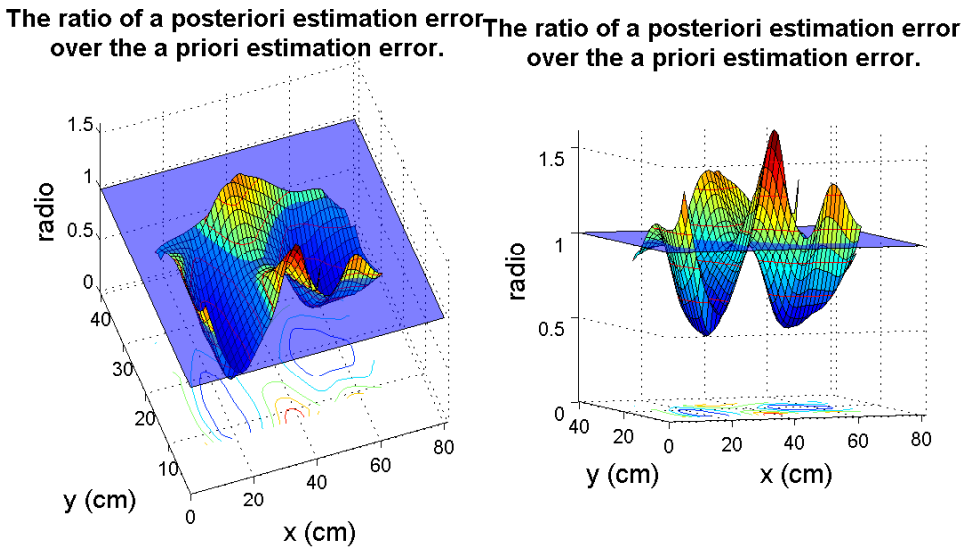
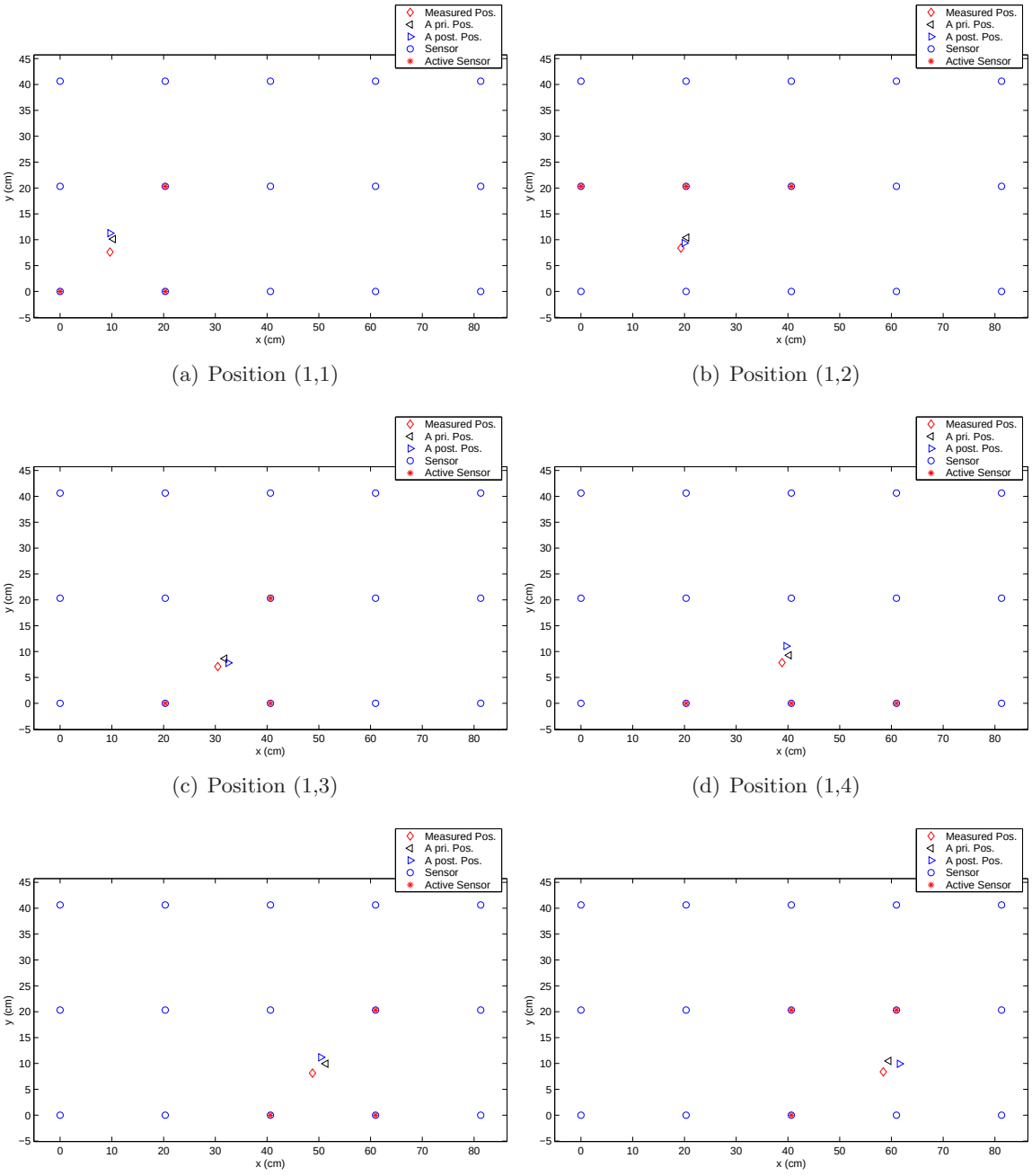


Fig. 3.25: A "before-and-after" comparison of our sensor selection testbed: ratio of the a posteriori estimation error (after sensor selection) over the a priori (before sensor selection) estimation error.



(a) Position (1,1)

(b) Position (1,2)

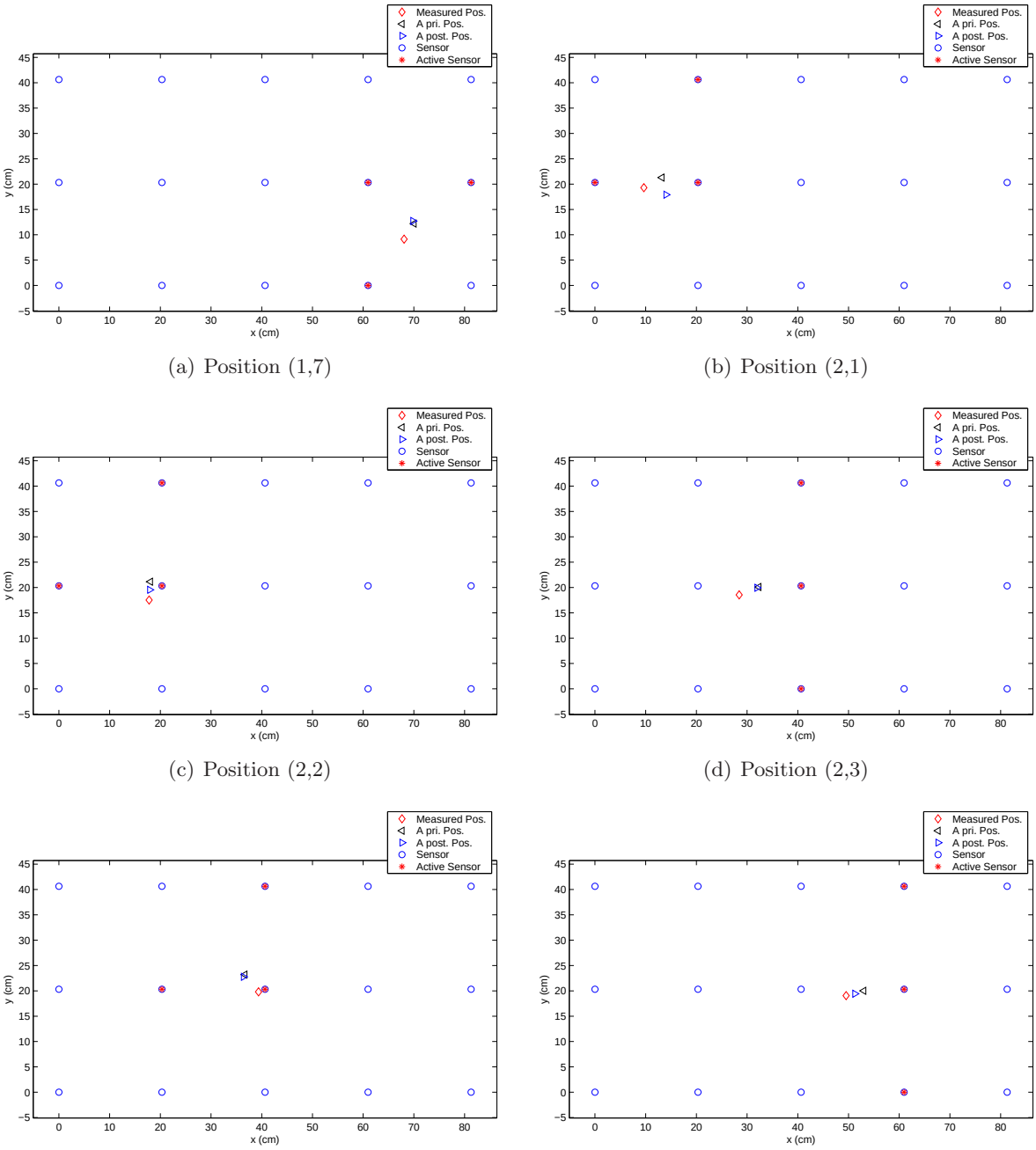
(c) Position (1,3)

(d) Position (1,4)

(e) Position (1,5)

(f) Position (1,6)

Fig. 3.26: Hardware testing results for sensor selection: positions (1,1) to (1,6).



(a) Position (1,7)

(b) Position (2,1)

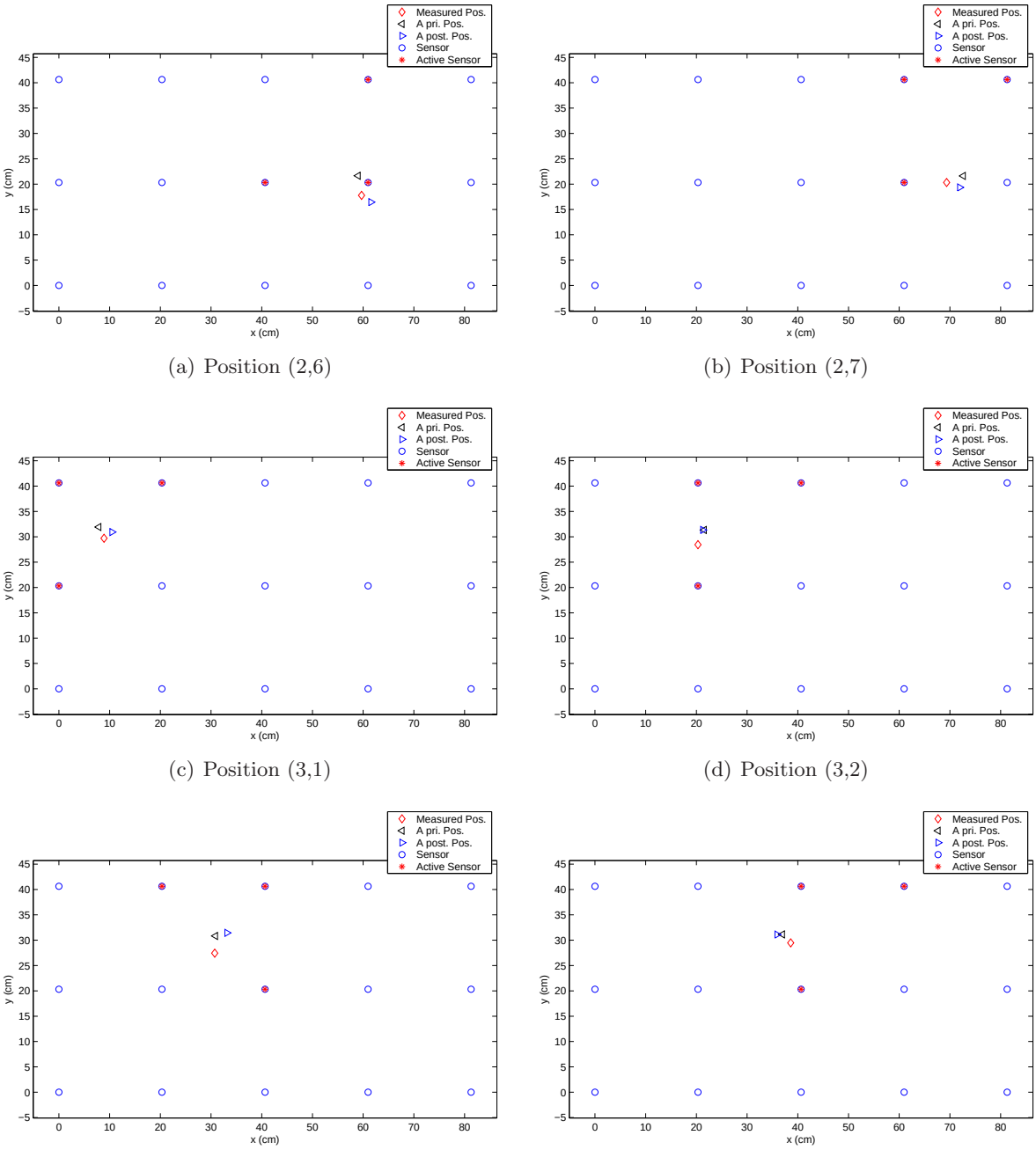
(c) Position (2,2)

(d) Position (2,3)

(e) Position (2,4)

(f) Position (2,5)

Fig. 3.27: Hardware testing results for sensor selection: positions (1,7) to (2,5).



(a) Position (2,6)

(b) Position (2,7)

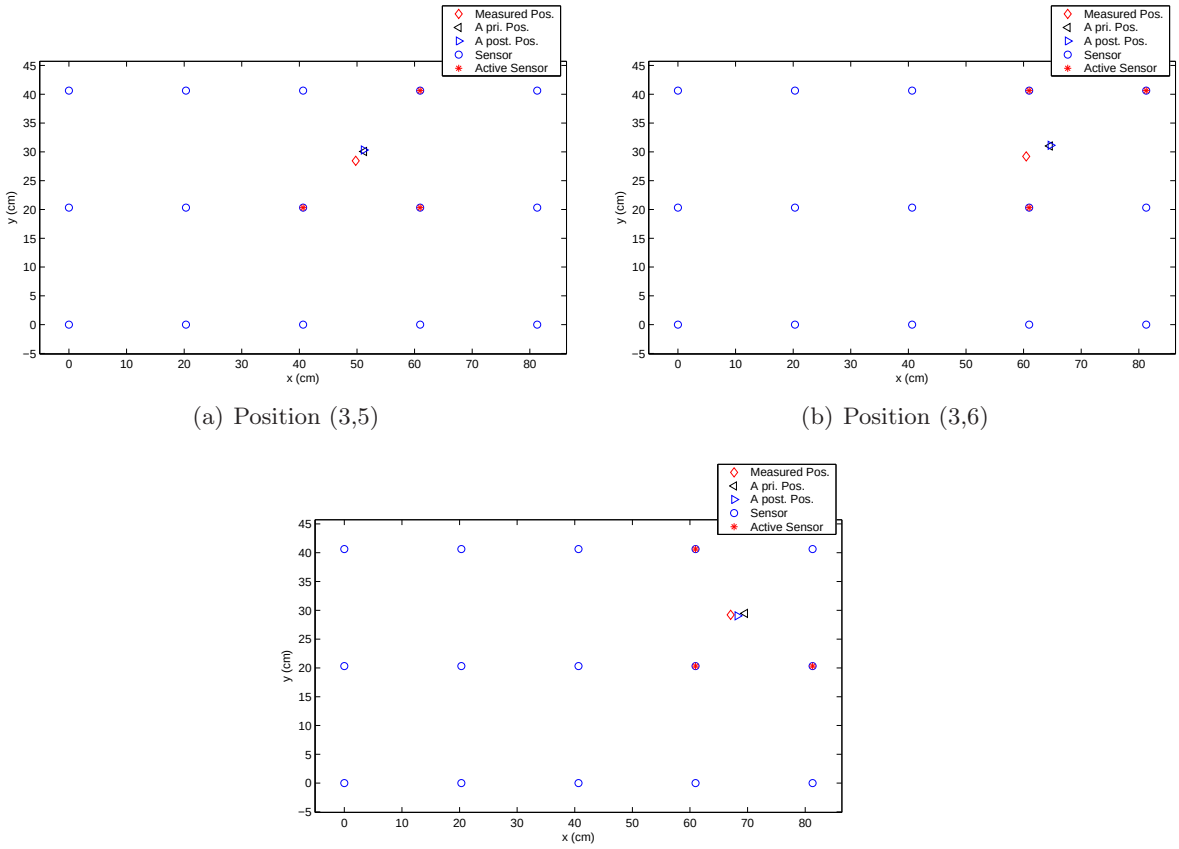
(c) Position (3,1)

(d) Position (3,2)

(e) Position (3,3)

(f) Position (3,4)

Fig. 3.28: Hardware testing results for sensor selection: positions (2,6) to (3,7).



(a) Position (3,5)

(b) Position (3,6)

(c) Position (3,7)

Fig. 3.29: Hardware testing results for sensor selection: positions (3,5) to (3,7).

3.4 Discussion

3.4.1 Remarks on the Speed and Memory Requirements

The COSS algorithm is simple, fast, and memory efficient.

Comparing to other related methods, the COSS algorithm is simple and fast. We use the Profile tool from Matlab and test the speed of our Matlab version COSS algorithm. On a 3Ghz Pentium 4 PC with 1Gb memory, it takes the COSS algorithm around 8 ms to optimize a network with 60 sensors. In our testbed, the COSS algorithm is implemented by the C# language and execute in real-time. After the sensor nodes submit their data to the sink, no delay can be observed before the sink selects the sensors based on the COSS and broadcast the results to the sensor nodes.

It is reported in [48] that the sensor selection based on B&B integer programming method "allows us to schedule up to 50-60 sensors typically in the order of seconds." In [100], the sensor selection patterns are computed off-line and stored in a look-up table. That paper says that "in an application where less than six sensors must be allocated per target, the $O(n^5)$ running time may be prohibitive for real-time implementation."

The COSS algorithm requires memory to store $\nabla_{\mathbf{q}}\Psi$, M, and $\mathbf{a}_i$. Thus, it requires memory for $2mn + m^2$ float or double variables, where m is the number of parameters and n is the sensor number. This is a small number with respect to the grid-based methods. Since m is small ($m = 2$ or 3 for tracking problems) the required memory is not much. Considering grid-based Bayesian or entropy approaches, those methods need memory to store k^2 or k^3 cells, where k is the number of cells in each dimension.

3.4.2 Comments on Non-Gaussian Noises

The formation of FIM in (2.4) is derived under the assumption that the sensor noise is Gaussian. The derivation of the FIM is presented in Appendix B. What if the noise is not Gaussian? In practice, non-Gaussian noise is not unusual. The question can be answered from theoretical analysis and hardware experiments.

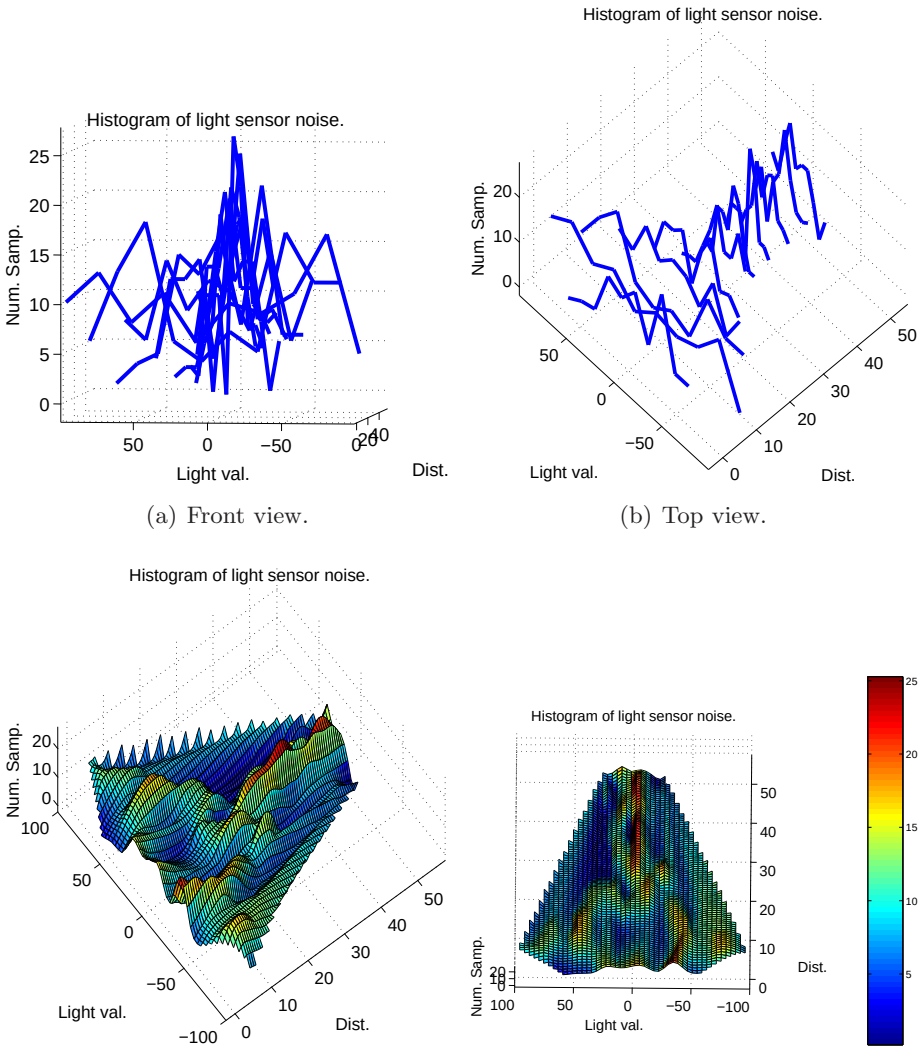
From the theoretical aspect, the side effects of the non-Gaussian noise can be compensated by larger number of samples. There are connects between the Gaussian and

non-Gaussian noises. It is claimed in [109] that "since the covariance error matrix for independent Gaussian distributed errors is the same as the asymptotic covariance matrix of any distribution [114] it is usual to assume that the errors are Gaussian." Comparing Gaussian and Non-Gaussian is a rich topic. Further discussion is out of the scope of this dissertation. Based on the aforementioned result, if there are enough number of samples, the solution of minimizing the covariance matrix of independent Gaussian noises applies to non-Gaussian noises as well. In practice, we can increase the number of samples if the estimation precision is not satisfied.

Under this context, it is valuable to test the algorithm on a physical testbed with non-Gaussian sensor noise. In fact, our experimental data indicates that the sensor noise is not ideal Gaussian. Based on the hardware experiments, we observed that the COSS algorithm is not sensitive to the non-Gaussian noise. However, under this condition, the a posteriori estimates are still more precise than that of the a priori estimates.

Hereby we study the distribution of the sensor noise. Figure 3.30 includes the plots of noise's histogram. During the experiments, Tmote Sky sensor nodes are placed under a halogen lamp at 11 positions whose ranges are uniformly distributed. 100 samples per distance are collected. One histogram can be drawn for each distance. For easy comparisons, the centers of those histograms are shifted to 0 and they are plotted in 3D figures. The 11 histograms are plots as 11 curves in Fig. 3.30(a) and Fig. 3.30(b). The histograms are also plotted by 3D surface in Fig. 3.30(c) and Fig. 3.30(d). The sensor data that collected at ranges closer than 30 cm have irregular histograms. The sensor noises are not always Gaussian. Fig. 3.31 is the same as Fig. 3.30 except under different lamps. All the data in Fig. 3.30 is collected under a halogen lamp, while Fig. 3.31 is scenario under a florescent lamp. It is easy to see that the non-Gaussian noise exists under both of the situations.

In summary, although the COSS algorithm is derived based on Gaussian noise, the method is also applicable to non-Gaussian noises. Although some error may be introduced. The claim is supported by existing theory [109] and our extensive hardware experiments.



(a) Front view.

(b) Top view.

(c) Side view.

(d) Top view.

Fig. 3.30: 3D plots of the sensor noise histogram. (under a halogen lamp)

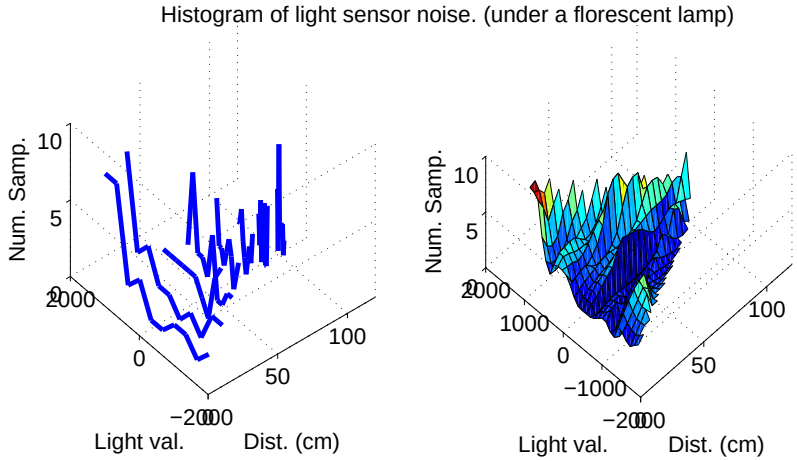


Fig. 3.31: 3D plots of the sensor noise histogram. (under a florescent lamp)

3.4.3 Relationships with Geometric Approaches

The sensor selection method in [100] is based on pure geometric approaches. Therefore we compare our algebraic approach with the geometric approaches.

Figure 3.32 illustrates the rough ideas of the two methods. The geometric method is depicted in Fig. 3.32(a) and Fig. 3.32(b). The rest plots are associated with the algebraic method. A camera-like sensor studied in [100] can be characterized as a cone in Fig. 3.32(a). The angular estimation error on the target is the same as the angle of the cone. At least two sensors are required in order to estimate the position of the target, i.e., (x, y). Figure 3.32(b) depicts the example when two sensors locate the position of a target. The estimation error is defined as the overlapping region [2] of the cones, since the estimate on the position of the target must fall inside both of the cones. As more sensors introduced, the overlapping region is a convex polygon [100]. It is proved in Lemma 4 of [100] that no more than 6 sensors are required in order to reduce the area of the overlapping region to be no more than twice of the minimum area.

Consider the algebraic approach using our formulation. Assuming the real position of the target is $(0.5, 0.5)$. In Fig. 3.32(c) the only one sensor's measurement is $\theta_A = 45°$,

[2] It is called the minimum enclosing parallelogram (MEP) in [100].

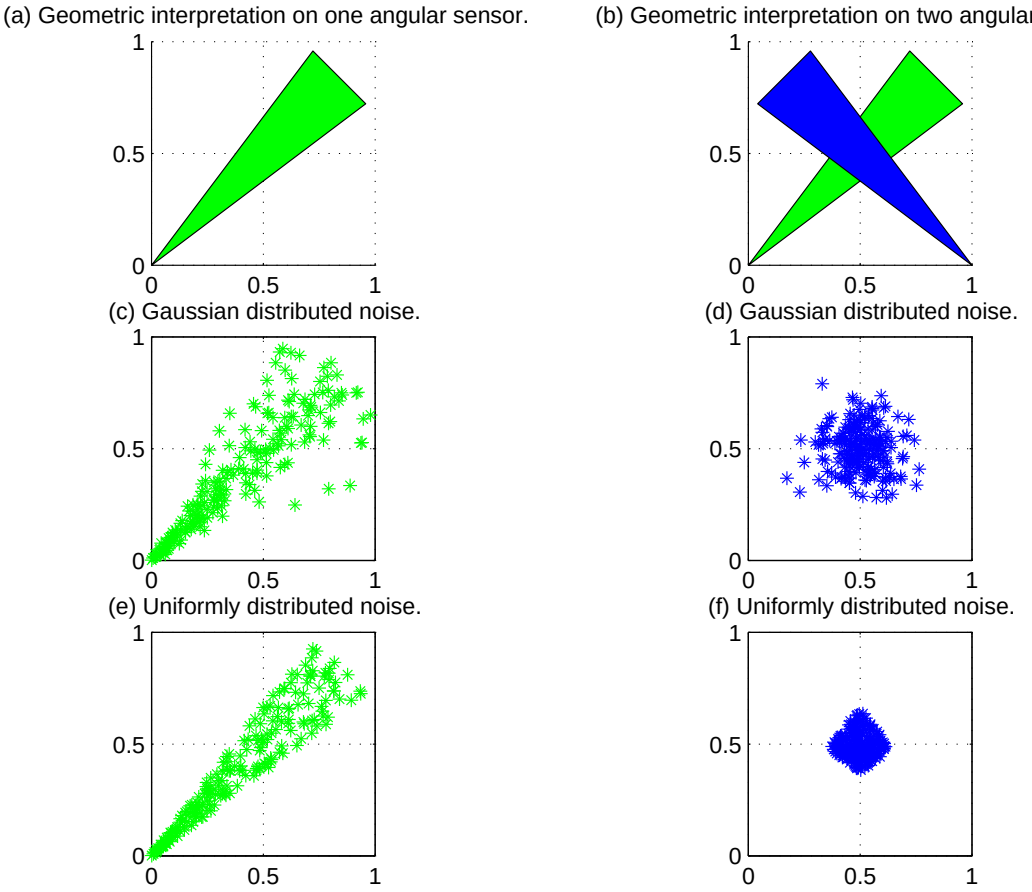


Fig. 3.32: Comparing the algebraic and geometric interpretations on estimation errors.

which is corrupted by additive Gaussian noise v, and $v \sim \mathcal{N}(0, \sigma)$. The sensor's position is at $(0, 0)$. The possible positions of the target, based on θ_A only, are indicated by the stars.

There are two angular sensors in Fig. 3.32(d). Their positions are $(0, 0)$ and $(1, 0)$, and their nominal angular measurements are $\theta_A = 45°$ and $\theta_B = 135°$, respectively. θ_A and θ_B are contaminated by independent Gaussian noise. After simple computations we have:

$$\begin{aligned} \hat{x} &= \frac{-\tan(\theta_B)}{\tan(\theta_A) - \tan(\theta_B)}, \\ \hat{y} &= \frac{-\tan(\theta_A)\tan(\theta_B)}{\tan(\theta_A) - \tan(\theta_B)}. \end{aligned}$$

In other words, one position, $(\hat{x}, \hat{y})$, can be estimated from one tuple of the angular measurements, $\{\theta_A, \theta_B\}$. Each star in Fig. 3.32(d) is associated with an estimated position, i.e.,

$(\hat{x}, \hat{y})$. It is seemed that the stars roughly fall inside an ellipse, which can be parameterized by a 2×2 covariance matrix. Since the noise in Fig. 3.32(d) is Gaussian the envelop of those stars are more like an ellipse than a polygon. If more sensors are introduced, the position can be estimated by LS or WLS methods, and the size of the covariance is unchanged. Refer to (3.7) for details. For our algebraic approach, the estimation errors are always measured in ellipsoids whose dimensions are the same as that of the parameters under estimation.

Easy to see that the ellipsoid area with starts in Fig. 3.32(d) is an approximation to the overlapping polygon region in Fig. 3.32(b). Under certain conditions, the optimization on the ellipsoids is the same as the optimization on the polygons. Remind, [109] suggests that independent Gaussian distributions can be used to estimate the covariance matrix of any distribution.

Take a look at Fig. 3.32(e) and Fig. 3.32(f), which are the cases when the noises are subjected to uniform distributions. Except that the noise is uniformly distributed, Fig. 3.32(e) is the same as Fig. 3.32(c). The stars in Fig. 3.32(f) are estimated target positions. The envelop of those stars is the same as the polygon in Fig. 3.32 (b). Since the noises are uniform, the algebraic method in Fig. 3.32(f) is comparable to a Monte Carlo approach to compute the polygon in Fig. 3.32(b). As aforementioned, under proper conditions, the optimization on the covariance matrix of the stars in Fig. 3.32(d) is the same as that in Fig. 3.32(f). The similarity between Fig. 3.32(d) and Fig. 3.32(b) is obvious. Thus, the difference between Fig. 3.32(b) and Fig. 3.32(d) may not as significant as they looked.

The interesting fact is that the observations from the two different approaches are in consistent with each other. It is observed in [100] that "the estimates obtained by the four best sensors are as good as the estimates obtained from all sensors. Hence, the lifetime of the network can be significantly increased by restricting the number of active sensors to a small number without losing too much from the quality of the estimates." Follow the algebraic approach, we conclude that no more than $m(m+1)/2+1$ sensors are good

enough to provide the optimal estimate, for the worst cases. That is, when the sampling rate optimization is not optimized. (For the optimal solutions, $m(m+1)/2$ sensors are required. Check Section. 3.2.5 for the details.) When a target is in a 2D domain, $m = 2$, and the $m(m+1)/2+1$ is exactly four! Despite of the similarities, there are some difference between the two approaches:

- The polygon representations of the geometric approach may be more precise than the ellipsoid of the algebraic approach, for example, when the sensor noise is uniformly distributed.

- It is easier to consider high-dimensional systems by the algebraic approach. For example, if there are 4 or more parameters need to be estimated, it is not easy to draw overlapping polygons and compute their areas. For the algebraic approach, the change is trivial. Only adjust the size of the matrices.

- Replacing different cost functions in the geometric method (currently there is only one cost function) is not as easy as that in the algebraic method. This is explained in the following paragraphs.

In the remark of Definition 3.2.3, several optimality criteria are listed. They all have geometric interpretations. Hereby we explain them under the scenario in Fig. 3.32. The output of the D-optimality is proportional to volume of the covariance matrix, i.e., the area of the star clouds in Fig. 3.32(d) and Fig. 3.32(f). The geometric meaning of the D-optimality criterion is the same as the area of the overlapping region in [100]. The D-optimality is the most widely used criterion and its advantages are described after the Definition 3.2.3. However, it is not perfect. For example, it is possible that the volume of a covariance matrix is small but the positioning error is significant. For example, when the confidence ellipsoid is very tall or wide. Under those cases, the A-optimality, which indicates the positioning error measured in Euclidean distances, or E-optimality, which is proportional to the maximum diameter of the confidence ellipsoid, are desirable options. Since E- and A-optimality criteria are not invariant to linear transforms, we may need to

scale measurements into common units before applying the optimality criteria. For more discussions on other optimality criteria, such as the C-, T-, Tuning, and MV criteria, refer to [109, 115].

3.4.4 Entropy-based Method

The so called entropy-based methods in this dissertation can be interpreted intuitively: The sensor data which enhance the certainty on the estimate of the target's position is useful, and the sensors whose data is useful should be selected. The theory of Shannon entropy and Bayesian theory play central roles for some sensor selection methods [38, 49, 104, 105, 116]. This approach is also referred as the information theoretic method [105].

Roughly speaking, the Shannon entropy is comparable to the covariance matrix of the COSS. The LS fitting in COSS method plays the same rule as that of the Bayesian theory in the entropy-based method.

The IDSQ algorithm [104] is an example of the entropy-based methods. In this method, the belief on the position of a target is updated according to sequential Bayesian filtering. Based on the measurement of information, i.e., entropy, the sensor within the vicinity of estimated target position and provides the "most information" is selected. A heuristic interpretation on the entropy is as the follows: The selected sensor have the largest number of bits to contribute to the estimation on the target's new position. Comparing to IDSQ, the COSS follows a different logic: The gradient in Fisher information is a metric of sensitivity. The valuable information contained in sensor data is measured by the sensitivity of the sensor readings with respect to the physical parameters under observation. If a sensor is "sensitive" to the parameter, its reading changes significantly due to the perturbations on the parameter. In plain words, the COSS method selects sensors that are most sensitive to the target's position (or other parameters). The Bayesian filtering in IDSQ is similar to the LS in COSS, in the sense they both update estimates based on sensor data. Of course, they are also significantly different since the target's dynamics can be considered under the framework of Bayesian filtering, but not under the LS.

Table 3.2 compares the IDSQ and the proposed COSS algorithms. As we see, most entries of them are complementary. The followings are some remarks on the table.

The IDSQ is fully distributed, which is an advantage over the COSS method. Although the COSS has potential to be implemented in a fully distributed fashion, most of the computation of the COSS method is currently carried out by the sink, and the sensor nodes are responsible for a small part of the computation. Note that, the distributed D-optimization and LS fitting have been developed [44, 46].

The dynamics of the target are considered by the IDSQ method, due to the sequential Bayesian filter. Currently, the dynamics are not included in the COSS, and it is our future work is to incorporate distributed Kalman filter-like algorithms within the COSS, similar to [117, 118]. Kalman filter can be consider a special Bayesian filter. A significant differences between the two is that the Kalman filter is not grid-based algorithms, but the Bayesian filter is. Since the COSS method is also not grid-based, it is compatible with the Kalman filter. The original Kalman filter is centralized and subjected to Gaussian noise. Currently, there are distributed Kalman filters [118] and the Gaussian noise is not mandatory.

One obvious difference is that the IDSQ is a grid-based algorithm while the COSS is not. The IDSQ method requires several 2D meshes to store the probability mass functions. The higher the resolution, the more the grids are required. On the contrary, the COSS only requires two arrays to store the sampling rates and sensitivities of each sensor. The length of the arrays is determined by the number of sensors. It may be faster and less demanding on the memory.

Both methods accept the non-Gaussian sensor noise, but due to different reasons. The IDSQ method is grid-based, thus nonparametric. Any probability distribution can be represented by the grids. The COSS is a parametric non-grid-based method and deduced based on Gaussian noise. However, as Section 3.4.2 presents, non-Gaussian noise may be acceptable, with some scarifies on the estimation precision.

The key feature of the COSS method is that it optimizes the estimate with a small number of sensors. That is, the estimation error is close to CRLB and the number of

selected sensors is determined by the Carathéodory's theorem. For the IDSQ, the most "informative" one sensor is selected. If all the sensors are selected, the Bayesian method (in the IDSQ) provides the "maximum a posteriori" (MAP) estimate on the parameter. The MAP estimate is close to LS estimate under many conditions [109]. Intuitively, if only one sensor is selected, the estimate may not better than the MAP estimate.

Finally, both methods have been tested by hardware. The IDSQ testing scenario is outdoor vehicle tracking. The COSS has been test by the aforementioned testbed.

3.4.5 Discussions on Correlations of Sensor Data

One of the assumptions in this chapter is that the sensor noises are independent. Such an assumption is important for the analysis in Section 3.4.2 and Section 3.4.3. In the following paragraphs, we argue that such assumption is reasonable.

In the following paragraphs, we firstly present our experiment data and draw conclusion from the data that the sensor noises are independent. Then our simulations indicate that the dependency among noises can be canceled by properly designed hardware systems.

The dependency of the sensor noise can be measured by the correlations of the sensor data. Remind that the correlation between random variables, X and Y, is measured by the correlation coefficient $\rho_{X,Y}$.

$$\rho_{X,Y} = \frac{E\{(X-\mu_X)(Y-\mu_Y)\}}{\sigma_X \sigma_Y},$$

Table 3.2: Comparisons on IDSQ and COSS Methods.

Properties	**IDSQ**	**COSS**
Target dynamics	yes	no
Fully distributed	yes	no
Grid-based method	yes	no
Computation cost	high	low
Memory requirements	high	low
Accept non-Gaussian noises	yes	yes, but some precision is sacrificed
Estimation precision in theory	unknown	close to CRLB
Number of selected sensors	one at a time	guided by the Carathéodory's theorem
Hardware experiment	yes	yes

where μ_X, μ_Y are the mathematical expectations of the random variables, and the σ_X and σ_Y are the standard deviations. Thus, if the sensor noises are v_X and v_Y, we have $v_X = X - \mu_X$ and $v_Y = X - \mu_X$. If and only if the sensor noises are independent, the following equation holds:

$$E\{v_X v_Y\} = E\{v_X\}E\{v_Y\}.$$

One may think that the correlation between sensor data increases as the inter-sensor distance reduces. This intuition is wrong. When the sensors are close to each other, their measurements are close. However, those sensor data may or may not be correlated. Fig. 3.33 is a visual interpretation on the claim. Suppose two sensors are so close to each other that their measurements, X and Y, have the same expectation (mean), or $\mu_X \approx \mu_Y$. Figure 3.33 shows that X and Y may or may not be correlated. In Fig. 3.33(a) there are three point clouds, representing three set of measurements. The centers of the clouds, i.e., the expectations, are $(0, 0)$, $(1, 1)$, $(2, 2)$. For sensor selection purposes, one out of the two sensors should be selected, since their mean measurements are the same. In Fig. 3.33(b), X_i and Y_i are the ith measurement of X and Y, respectively. The contour in Fig. 3.33(b) represents the values of the correlation coefficients between X and Y. As Fig. 3.33(b) indicates, X_2 and Y_2 are not correlated. X_1 and Y_1 have positive correlations, and X_3, Y_3 have negative correlations. In summary, no conclusion on the correlation can be drawn from the fact that $\mu_X \approx \mu_Y$.

So, when the sensor data is correlated? In the context of sensor networks, we argue that *common impact factors* introduce correlations between sensor data. The so called common impact factors are the phenomena that influence many sensor measurements at the same time. A contradictory term is *local impact factors* which only affect one sensor's reading. Examples of the common impact factors include ambient light, room temperature, lamp flashing pattern, which will be discussed soon. Local impact factors include sensor node ADC quantization errors, battery imperfectness, sensor noise, etc. Why the common impact factors introduce correlations? It is due to the fact that the common impact factor perturbs the readings from different sensors with certain predictable trends. For example,

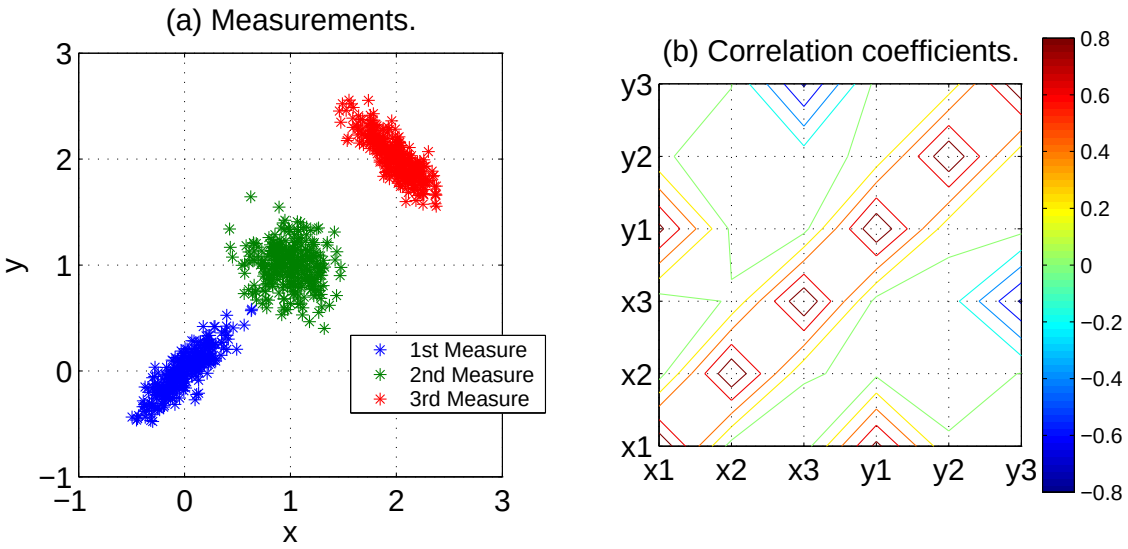


Fig. 3.33: Illustration on correlations.

if X is increased due to the brighter ambient light, Y should also increase due to the same ambient light. This case is similar to the 1st measure (at the position (0,0)) in Fig. 3.33, where Y tends to increase while X increases. Then, as aforementioned, X and Y are correlated. Comparing to the local impact factors, such as the quantization errors, there is no predictable trends among different sensors. When a sensor takes one measurement on the light value, some error is introduced due to the quantization of the ADC chip on the particular sensor node. No other sensors are affected by the same ADC chip. Hereby, no common trend between sensor data exists. Thus, there is no correlations between the sensors. This case is similar to the 2nd measure in Fig. 3.33.

Let us exam the statements based on our experimental observations. In addition, we give an example on how to cancel sensor data correlations, if they exist, based on our experiments. Fig. 3.34(a) is same as Fig. 3.2. From these figures, we conclude that under halogen lamps, the light sensor measurements are statistically uncorrelated, since the absolute values for the coefficients of cross-correlations are close to zero. If we had chosen a florescent lamp, the degree of correlation among sensors is higher.

Fig. 3.34(b) is the correlation coefficients for the light sensors on Tmote Sky nodes under a florescent lamp. This figure is also plot based on our experiment data. Unlike

the halogen lamps or incandescent lamps, which do not flash, the florescent lamp flashes at the same frequency as the power line. (60Hz in the US.) Comparing Fig. 3.34(a) and Fig. 3.34(b), it is easy to see that the correlation of the sensors under the florescent lamp is more obvious. The mean absolute value of the correlation coefficients and the cross correlation coefficients of data under the florescent lamp is larger than that under the halogen lamp. From this aspect, the halogen lamp is better than a florescent lamp for our application.

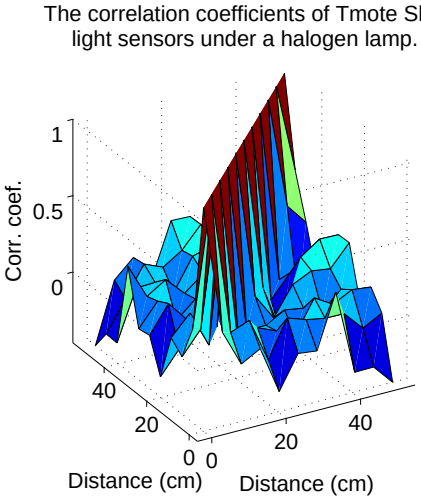
We also simulate flashing lamps based on the experiment data. In addition to show that the flashing lamp is a common impact factor, the simulation also indicates that there could be many types of correlation patterns. Fig. 3.34(c) and Fig. 3.34(d) are plotted based on the simulation results. The scenario in Fig. 3.34(c) is as the follows: The flashing frequency is as half of the sampling rate of those sensors. At the time instances when the lamp is off, the sensors measure the ambient light, which is a constant. While the lamp is on, the measurement is the addition of the ambient and the lamp's light. If $\mathbf{d}_E$ is the data collected in experiment, and $\mathbf{d}_A$ is the simulated flashing light, then $\mathbf{d}_A$ is defined as:

$$\begin{aligned} \mathbf{d}_A[2k] &= \mathbf{d}_E[k], \\ \mathbf{d}_A[2k+1] &= c_A, \end{aligned}$$

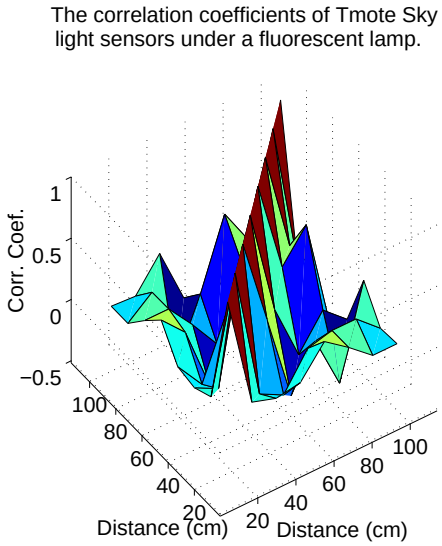
where c_A is a constant to simulate the ambient light, and k is the trail number. From Fig. 3.34(c) we see that the correlation among the sensor data are significant: Most of the correlation coefficients are either close to 1 or -1. Fig. 3.34(d) is the simulation for time-varying ambient light: The intensity of the ambient light follows a sinusoid signal, thus time-varying. The illumination of the lamp is steady. The readings from the sensors are the summation of the ambient light and the lamp's light. If $\mathbf{d}_B$ is the simulated varying light, the following holds:

$$\mathbf{d}_B[k] = \mathbf{d}_E[k] + \mathbf{s}_A[k],$$

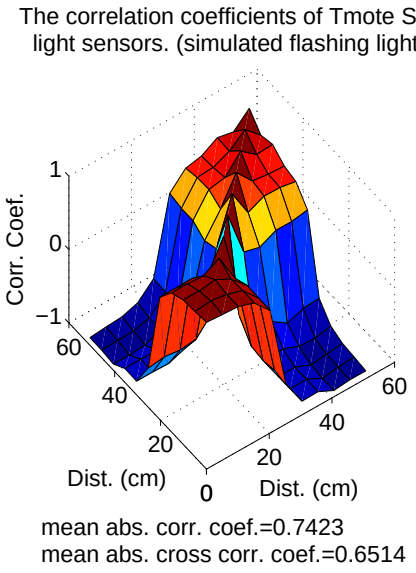
where $\mathbf{s}_A[k]$ is a sinusoid function with respect to k. From Fig. 3.34(d), we see that the correlations among sensor data are also very high. However, the correlation pattern between Fig. 3.34(c) and Fig. 3.34(d) are different.



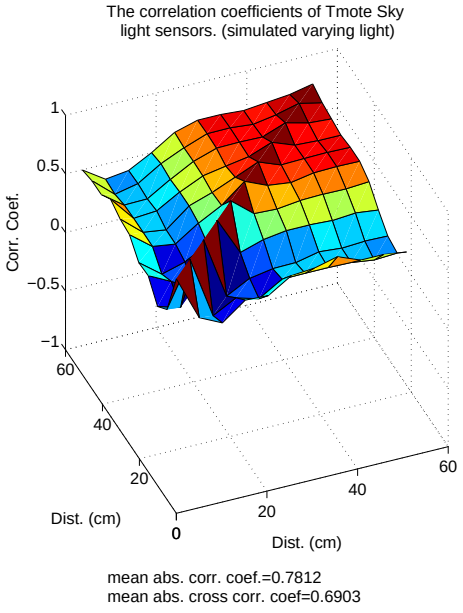
(a) Exp. data for a halogen lamp.



(b) Exp. data for a florescent lamp.



(c) Simulated flashing light.



(d) Simulated varying light.

Fig. 3.34: Correlation coefficients of sensor data.

Comparing Figs. 3.34(b),(c), and (d), we see that the correlation patterns could be significantly different and irregular. Since the correlation pattern changes as the common impact factor changes. It is not likely to present all the possible correlations by several groups. The reasons are as the follows:

- It is unacceptable for an engineering project to have unknown common impact factor. In stead of studying the statistics characteristics of the impact factor, the engineer must find out the source of the impact. Otherwise, the project is unreliable, since its success depends on some uncontrolled factor, which may be changed at any time.

- After finding the common impact factor, we can cancel the factor by either hardware improvement or low-level filtering. Take our sensor selection testbed as the example, the dependency of the sensor noise can be canceled if the florescent lamp is replaced by a halogen lamp, or a software band stop filter which rejects 60Hz signals.

- As aforementioned, due to the complexities of the correlation pattern, it may be difficult to develop a generic sensor selection method for a large class of correlations. After understanding the physics of the common impact factor, it is easier to reject the correlation at the lower level. Still take the flashing florescent lamp as the example, it is much easier to replace a lamp, rather than design a D-optimization method which does not require the independent sensor noises.

In summary, it is a reasonable to assume that the sensor noises are independent. If the raw sensor data is correlated, it may be desirable to cancel the correlation at the lower level.

3.4.6 Comments on Networking

Traditional networked communication systems are designed based on stacks, or layers. The layered structure of our sensor selection testbed is not presented by the system architecture in Fig. 3.3. For comparison purposes, the layered model of our sensor selection testbed is plotted in Fig. 3.35.

One of the key features of this model is that the physical model plays an important role, as if a layer in the communication stack. Another key feature is that the parameter of communication layers is tuned by the observer layer. Thus the observer not only transmits and receives information from the communication stack, but also controls the stack.

As it can be seen in Fig. 3.35, the model of the physical world is required for the design of the observer, which is essentially the application layer of other communication models. The feature of this model is that a parameter of the communication stack, the transmission rate, is tuned by the observer. Remind that if a sensor is not selected, its transmission rate is zero. If it is selected, the rate is a constant. Since the observer is model based, the parameter is in fact determined by the measurements on the physical world.

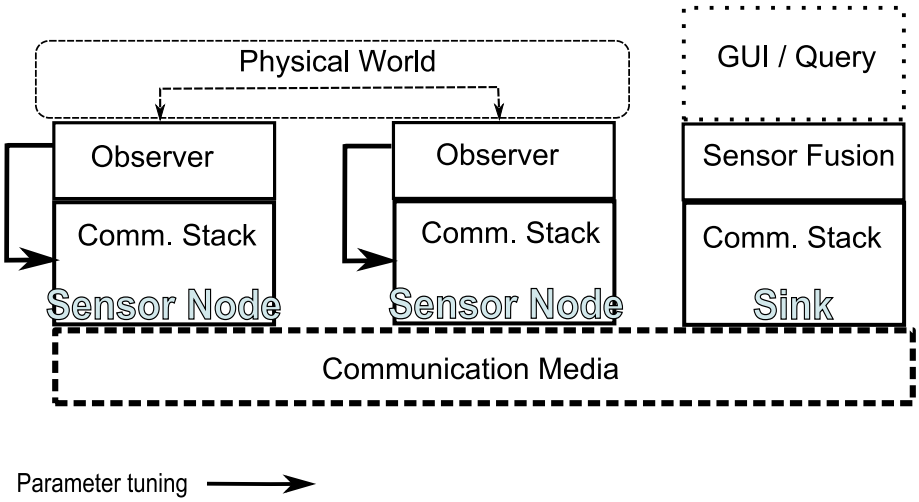


Fig. 3.35: Communication stacks of the COSS method.

3.5 Chapter Summary

In this chapter, we proposed a COSS algorithm to solve the sensor selection problem. The algorithm not only selects the minimal number of sensors that allowed by the Carathéodory's theorem, but also pushes the estimation errors closer to the theoretical lower limit, i.e., the Cramér-Rao lower bound. After extensive simulation and hardware experiments, we conclude that the approximations are reasonable for engineering practices.

Our theoretical analysis reveals the existing of a class of sensor selection methods that are similar to the COSS method. They are named implicit optimal sensor selection method in this chapter.

In future, we will develop fully distributed COSS and implement it on low-cost sensor nodes for real-time target tracking.

Chapter 4
Design and Optimize Localization Systems for Wireless Sensor Networks

4.1 The Motivation and the Problem

Localization is a fundamental function of WSNs. Many higher-level functions, such as routing [119], either depend on localization or work better if the sensors' positions are available. Because of its importance, WSN localization has been discussed intensively. For example, a decentralized WSN localization system based on signal strength of Radio Frequency (RF) signals is proposed, namely MoteTrack [13]. In [120], a Time-Of-Flight (TOF) based acoustic localization system called Cricket Board is presented. Currently, in the de facto WSN hardware standard, IEEE 802.15.4, localization is not specified [12], and the widely used CC2420 chip [121] only supports Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI) as the measurements of distances. Some researchers propose algorithms to locate sensor nodes with the current IEEE 802.15.4 standard [13, 122]. However, RSSI and LQI are proved to be unreliable as the distance indicators, especially for indoor environments [16, 123].

4.2 Localization Hardware

4.2.1 RSSI

This approach takes the RSSI of RF signals as the indicator of distance. This method is featured with virtually no additional hardware costs and little demands on energy. However, the method is also not precise. Comparing to acoustic TOF-based methods, the resolution of the RF RSSI-based methods is very limited. While the acoustic methods have centimeter level precision, the resolution for RSSI-based methods may be about several meters [13]. Due to multi-path fading effects of RF signals, there is no direct relationship between RSSI and the distance between transmitter and receiver [16], especially for

in door environments [13]. In addition, our experiments reveal that the RF signals also have strong time varyings [124]. This is yet another challenge for RSSI-based localization systems. To our best knowledge, no state-of-art RSSI-based method supports a resolution under 2 meters, which may not be ideal for many applications. In addition, based on our experiments, non-trivial tunings are required by RSSI-based systems.

4.2.2 AOA

Methods based on angle of arrival (AOA) [67, 104] are also used for sensor node or target positioning. Acoustic AOA measurement [104] is easier than RF AOA, since the latter method may require high-speed signal processing. Although AOA has been discussed in theory [67], it is not a common localization method for low cost sensor nodes, probably due to its hardware costs.

4.2.3 Acoustic TOF

This method measures TOF of acoustic signals, such as sonar signals [120]. This method may provide centimeter level precision. Comparing to RF TOF method, the measure on acoustic TOF is much easier, since the speed of sound is much less than that of the RF signals. However, RF signals have several advantages over the acoustic approaches.

- The speed of sound wave is not as stable as that of the RF signals. The speed sound wave is affected by the temperature and humility of the environments. Thus, the measurement on TOF of RF signals, although may be more difficult, could be more precise. For example, it is reported that sub-millimeter level precision can be achieved by RF TOF measurements [125].

- Ultrasonic sound wave is directional. The orientation of the acoustic device may affect the positioning results [120]. Lower frequency sound, although is omni-directional, can be heard by ears, thus may not be desirable. Since the RF signal can be emitted omni-directionally with proper antenna, the placement of RF TOF-based sensors are less restrictive [20].

- RF devices may be more energy efficient than the acoustic counterpart. A RF TOF device may require power of micro-watt level [126], while milli-watt level power may be required by sonar devices [120].

4.2.4 RF TOF Measurement

This method could be both accurate and energy efficient, however, it is also very challenging. We need a low-cost clock with very high precision. Currently, there are several solutions within this domain. The difference and relationship between our proposed method and existing methods will be discussed soon.

Because light travels at a speed of 3×10^8 m/s, we need a device equivalent to a counter running at 300 MHz, in order to get sub-meter level distance measurements. Although counters with such a speed are common at these days, they are too costly for WSN applications. Since typical clock speed of the processor on a sensor node is about 8 MHz to 16 MHz, it is obvious that the TOF can not be directly measured by the processors and a special hardware for TOF measurement is required. Several hardware implementation methods are listed below:

1. Application Specific Integrated Circuit (ASIC): Several ASIC chips with RF TOF measurement capability have been developed [121, 123]. Using modern technologies, we can even directly make counters running at 300MHz on the chip, and hence the TOF can be measured through brute force approaches. However, the development costs of even simple ASIC chips are very high. Although the cost per chip is the lowest among compared with other TOF solutions, the high initial development cost made this method unsuitable for many applications.

2. Field Programmable Gate Array (FPGA): In short, FPGA is a lower-cost replacement for ASIC. In [127], a FPGA along with a 100 Mhz physical clock is proposed to achieve a resolution that approaches to 4 Ghz clock. The speedup factor (4Ghz/100Mhz) is 40. The system is specially designed for FPGA and no processor is required. The TOF measure system in [128, pp79] has stand alone communication

system and is targeted for ASIC implementations, similar to the solution presented in [129].

3. Discrete components: There are several systems that measure TOF using commercial-out-of-shelf electronic components. For example, McEwan developed several TOF measurement devices using UWB (ultra-wide band) radios, such as the system in [130]. In order to increase signal to noise ratio (SNR) and enable low-cost TOF measurement, the UWB transmitter in [130] sends thousands of impulses to get one distance measurement.

The document of the next generation WSN protocol, IEEE 802.15.4a, indicates that new RF hardware which measures the TOF of RF signals may be used to improve the localization significantly [131]. Actually, the candidate technologies, UWB (Ultra Wide Band) [132] and Chirp-based localization [133], such as CSS [134] (Chirp Spread Spectrum), are not totally new and have been studies for ten years or more. Technically, both approaches have the TOF location as well as the communication capabilities and both of them have commercial products. The UWB was chosen by IEEE 802.15.4a as the location method, and both of the UWB and the CSS methods will be used for communication. From the aspect of localization, both of the two provide narrow peaks as the indicators of the TOA of the RF signals. Each echo from the multi-path effect contributes a peak after the LOS (Line Of Sight) signal peak is received. TDOA (Time Difference Of Arrival) can be precisely estimated. In short, both of the methods use TOF as the indicator of the distance.

4.3 The Proposed Phase-based Localization Method

4.3.1 Review on TDOA Localization Algorithms

Comparisons with the current solutions: TOF Method Fig. 4.1 and Fig. 4.2 are from a technical proposal [1] of the IEEE 802.15.4a standard group. The two way ranging (TWR) method in Fig. 4.1 is also called RTT in other references [68]. It is very intuitive:

node A sends a message to the node B, and the node B acknowledge the message. In the acknowledgement, node B includes it reply time, T_{Reply}. Thus, after measuring the time between sending the message and receiving the acknowledgement, node A subtracts the T_{Reply} and the TOF can be easily computed. That is

$$T_{OF} = \frac{T_1 - T_0 - T_{Reply}}{2},$$

where T_OF is the TOF.

The acknowledgement from the node B is undesirable due to the following reasons:

- It is likely that the number of mobile nodes, m, is much more than the number of the beacons, n. Assuming pear-to-pear communication is available, this RTT measurement requires at least $m + n$ messages in total, for each mobile node to get its position. That is, beacons broadcast first, which requires n message packets. Then, the mobile nodes reply and m packets are sent. Thus, $n-1$ packets are required for one "sink beacon" to collect the range information from rest $n-1$ beacons. This sink beacon has the capability to locate each mobile node. To broadcast the positions to each beacon, at least one more packet is required. Thus, at least $m + 2n$ packets are required, in total. If there are many mobile nodes, their positions cannot be encapsulated in one packet and broadcast from the sink beacon, hence more than $m+2n$ packets are required. This is a centralized method with the sink beacon takes all the computation.

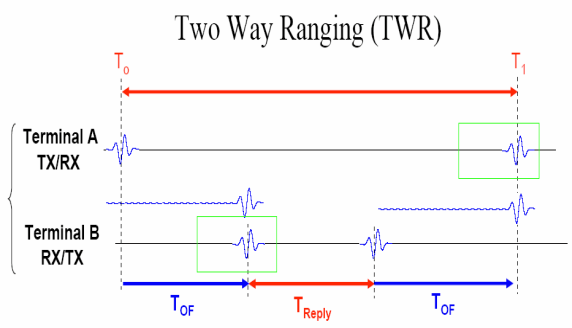


Fig. 4.1: Two way ranging method [1].

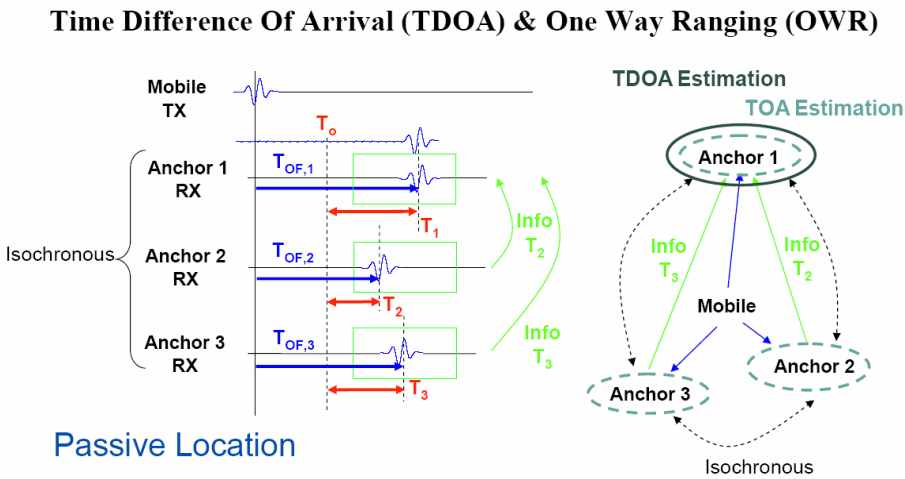


Fig. 4.2: One way ranging method [1].

- If the information flow is in the other direction. Mobile nodes broadcast to beacons and wait for the reply from the beacons, then at least $2m$ packets are required. This is a distributed method that the computation is uniformly distributed to each mobile node.

- For both the centralized and distributed method, the required number of communication packets is large if the number of mobile notes is big enough. The high demands of the RTT method on the communication also imply high demands on the energy.

- In practice, including T_{Reply} in the acknowledgement message may not be a precise method. From Fig. 4.1, the measurement of T_{Reply} requires the nanosecond-level time stamps when the acknowledgment is sent from node B. Thus, when node B is preparing the acknowledgement package, only the prediction, instead of the measurement, of T_{Reply} is available. Since in the time that a sensor node executes one command, the light may transfer hundreds of meters, T_{Reply} is hard to predict precisely to support cm level location accuracy. In the worst case, another acknowledgement should be sent from the mobile node in order to report the accurate measurement on the T_{Reply}. The bandwidth and energy is further sacrificed.

Another existing method is called one way ranging (OWR) method [135] which is also implemented in a commercial UWB localization system [136]. The method is a TDOA

(time difference of arrival) algorithm. From Fig. 4.2, it is seen that the only message is sent from the mobile node. The beacons (the anchor nodes in the figure) only listen to the mobile node. Thus, if the time intervals between the time when mobile nodes send the messages are large enough, the system is immune to the multi-path effects. However, the OWR requires the beacons to be synchronous. The nanosecond-level time synchronization is not trivial, especially if the synchronization signals are transmitted via lossy wireless channels. Note that the anchor nodes in [136] are connected by communication cables. In addition, the message flow from the mobile node still exists, which indicates that the energy and bandwidth cost of the OWR is about the same as that of the RTT and the localization is centralized. There is no significant improvement of the OWR over the RTT.

Following the conventions in RTT analysis, the OWR method requires at least $m+n+k$ packets for each mobile node to get its position, where k is the number packets that required for beacon synchronization. The number of packets is computed as the followings:

- Firstly, the m mobile nodes broadcast m packets.
- Secondly, $n-1$ beacons send $n-1$ packets to a sink beacon, which then has full knowledge on the range information.
- Thirdly, the sink beacon computes the positions of the mobile nodes and broadcast the information to the mobile nodes. At least one more packet is required.
- In total, we have $m+(n-1)+1$ packets, which equals to $m+n$.

Note that the OWR method is a centralized method. If the information flow of OWR is in the reverse direction, i.e., from beacons to the mobile nodes, the system is more suitable for WSNs, which has a large number of mobile nodes.

4.3.2 Problem Formulation

4.3.2.1 Design Considerations

It is desirable if a RF TOF-based localization system for WSNs has the following features:

- Distributed: The positions of each sensor node should be computed by the sensor nodes themselves, thus the burden on the communication and central computing unit is relaxed.

- Asynchronous: The TOF of RF signals are too small that the requirements on the precision on synchronization is very high, which may not be possible without wired connections or atomic clocks. In order to reduce the hardware costs, asynchronous localization systems are highly desirable.

- Energy efficient and hardware costs: The desirable system should be less expensive in both energy and hardware costs.

4.3.2.2 Phase of Arrival

We introduce the term Phase Of Arrival (POA) to represent a measurement that close to the TOA (Time Of Arrival), but not identical. The POA is motivated due to engineering considerations. Cost wise, the measurement on POA should be lower than the TOA. In addition, if a communication packet is lost, it is easy to handle the case in POA systems. The transmitter can simply retransmit the packet after one or more periods of the phase signal. The details are presented later in this chapter.

Assuming there is a "wrap up" counter, as the lower plot in Fig. 4.3. In the figure, the horizontal axis is the time, while the vertical axis is the phase. The phase signal is in a saw shape. The value of the counter starts from 0, increases by 1 at every time tick, arrives the maximum value, n_P, and start over from 0 again. For hardware implementation purposes, it is desirable if $n_P + 1$ is the power of 2. The upper plot of Fig. 4.3 shows that a signal arrives at certain time instance. The counter values at that time is called phase. The phase in this chapter is an integer within $[0, n_P]$. The cycling period of the counter

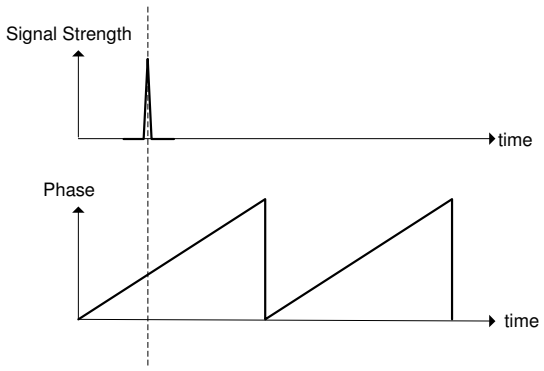


Fig. 4.3: Phase detection.

is called the period of the reference signal, t_R. Accordingly, there exists the frequency, f, and wavelength, λ, of the reference signal. The reason why the integer phase is introduced is due to the easiness of hardware implementation.

The POA measurement is close to that of the TOA. However, POA measurement requires a high-speed counter with fewer bits, which reduces the hardware costs. In cases when beacon packets are lost, the POA method retransmits the message packet after t_R. For TOA approaches, at least two high-speed counters are required for each node. For POA measurement, only one high-speed counter is required for each node. The length of the counter should be long enough such that within the time t_{OF} it is not wrap up. Although the chapter focuses on RF signals, the POA is a general idea, which is not limited to communication medium or communication protocols, such as such RF, or sound, or UWB, or CSS.

4.3.2.3 Phase-based Localization

The idea of phase-based localization is presented in Fig. 4.4. The top of the figure is an illustration on the topological relation of the beacons and the mobile node. Since the positions of the beacons are known, the distance r_{12} and its associated TOF t_1 are known. The distances r_{1A}, r_{2A} are unknown, so do their associated TOF t_2, t_3. The 3 charts, from top to bottom, at the bottom of the Fig. 4.4 are associated with the clocks on beacon 1, beacon 2 and the mobile node, respectively. The comments of Fig. 4.4 are the follows:

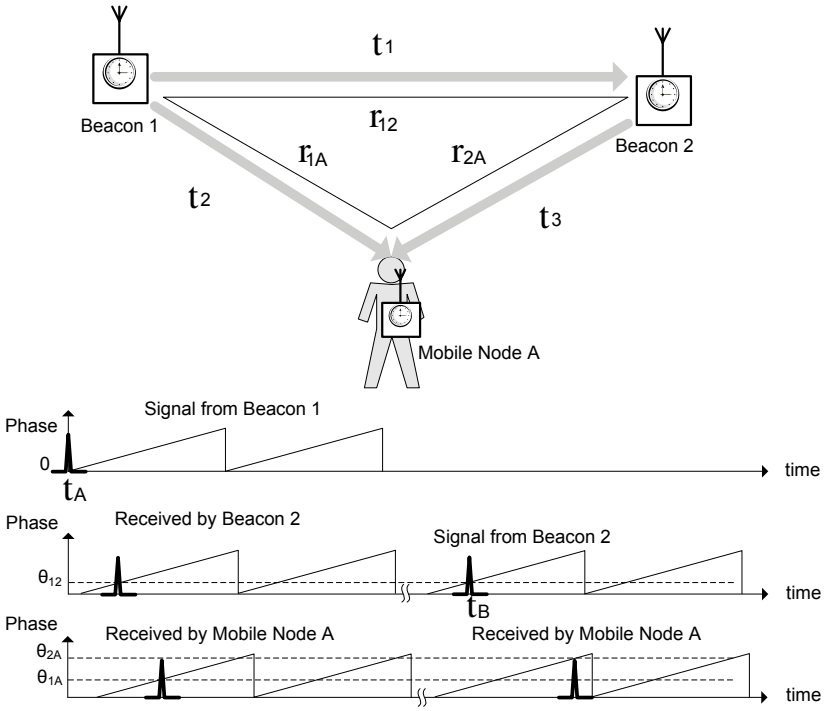


Fig. 4.4: Phase detection for TOF localization.

- The clocks on beacons and the mobile nodes are not synchronized. In the bottom of the Fig. 4.4, we see that the saw-like phases of the three clocks are not aligned.

- On time t_A, beacon 1 broadcasts an impulse message, which is received by beacon 2 and the mobile node shortly. The POA measured by beacon 2 is θ_{12} and that by the mobile node is θ_{1A}. Since the clocks are not synchronized, the initial phases on the three axes in Fig. 4.4 are different.

- Beacon 2 then must send an impulse message as soon as possible, with the same phase as it receives the impulse from beacon 1. If impulse is lost, e.g., communication collusion is detected while transmitting the impulse, then beacon 2 simply retransmits the impulse with a delay of one period of the reference signal. In the example shown by Fig. 4.4, on time t_B, beacon 2 broadcasts a message with the phase θ_{12}. The POA of this message by the mobile node is θ_{2A}. The delay between t_B and t_A is one or multiple periods of the reference signal. Comparing to Fig. 4.1, it is like that

T_{Reply} must be a non-negative integer multiplication of T_r, the period of the reference signal.

- Thus, 2 communication packets are required. One from beacon 1 and is broadcasted to beacon 2 and the mobile node. Another packet is broadcasted from beacon 2 to beacon 1 and the mobile node. Beacon 1 does not need to response to the second packet.

In the 2D domain, at least three beacons are required to provide a unique location on the mobile node. Fig. 4.5 illustrates the reason why synchronization is not required. Because beacon 2 receives and transmits the message from beacon 1 with the same phase, it is like a virtual "mirror" that reflects the message from beacon 1. Thus, the difference between θ_{1A} and θ_{2A} indicates the difference between $r_{12} + r_{2A}$ and r_{1A}. More precisely, if λ is the wavelength of the reference signal, we have

$$\frac{(\theta_{2A} - \theta_{1A})\lambda}{n_P} + k_1\lambda = r_{12} + r_{2A} - r_{1A},$$

where k_1 is a natural number. Remind that n_P is the maximum phase value. When λ is big enough, then no wrapping on the phase counter is possible. Thus, no ambiguity is possible by mapping the phase to the distance. That is

$$\begin{aligned} \frac{(\theta_{2A} - \theta_{1A})\lambda}{n_P} &= r_{12} + r_{2A} - r_{1A}, \\ r_{2A} - r_{1A} &= \frac{(\theta_{2A} - \theta_{1A})\lambda}{n_P} - r_{12}. \end{aligned}$$

That is, the mobile node is on a hyperbolic curve that subject to the following form

$$r_{2A} - r_{1A} = d_{12}, \tag{4.1}$$

where

$$d_{12} = \frac{(\theta_{2A} - \theta_{1A})\lambda}{n_P} - r_{12}.$$

In practice, estimation noise is always unavoidable. If we denote the noise associated with d_{ij} as e_j, the following equation holds

$$z_{12} = r_{2A} - r_{1A} + e_2, \tag{4.2}$$

where z_{12} is the measure of d_{12} based on sensor readings and r_{2A}, r_{1A}, e_2 are the unknown parameters. To locate the mobile node uniquely on a 2D domain, more independent equations in the form of (4.1) are required. Let us take the 3-beacon scenario as an example. The cases with more beacons are similar to this case. See Fig. 4.5, once installed, beacon 3 gets the message from beacon 1 when it broadcasts. Then, beacon 3 transmits the received signal just like beacon 2.

Repeat the above analysis and apply it to beacon 3. That is, replace beacon 2 in Fig. 4.5 by beacon 3. We have

$$\begin{aligned} \frac{(\theta_{3A} - \theta_{1A})\lambda}{n_P} &= r_{13} + r_{3A} - r_{1A}, \\ r_{3A} - r_{1A} &= \frac{(\theta_{3A} - \theta_{1A})\lambda}{n_P} - r_{13}, \end{aligned}$$

or

$$r_{3A} - r_{1A} = d_{13}.$$

Since beacon 3 and beacon 2 received the same broadcasting packet from beacon 1, only one more packet is required. If measurement noise is considered, the following equation holds

$$z_{13} = r_{3A} - r_{1A} + e_3,$$

where z_{13} is the measurement on d_{13}.

As an extension, if there are more than 3 beacons, the formulation for the POA method is as the follows:

$$\begin{aligned} r_{2A} - r_{1A} &= \frac{(\theta_{2A} - \theta_{1A})\lambda}{n_P} - r_{12}, \\ r_{3A} - r_{1A} &= \frac{(\theta_{3A} - \theta_{1A})\lambda}{n_P} - r_{13}, \\ &\vdots \\ r_{iA} - r_{1A} &= \frac{(\theta_{iA} - \theta_{1A})\lambda}{n_P} - r_{1i}, \\ &\vdots \\ r_{nA} - r_{1A} &= \frac{(\theta_{nA} - \theta_{1A})\lambda}{n_P} - r_{1n}. \end{aligned}$$

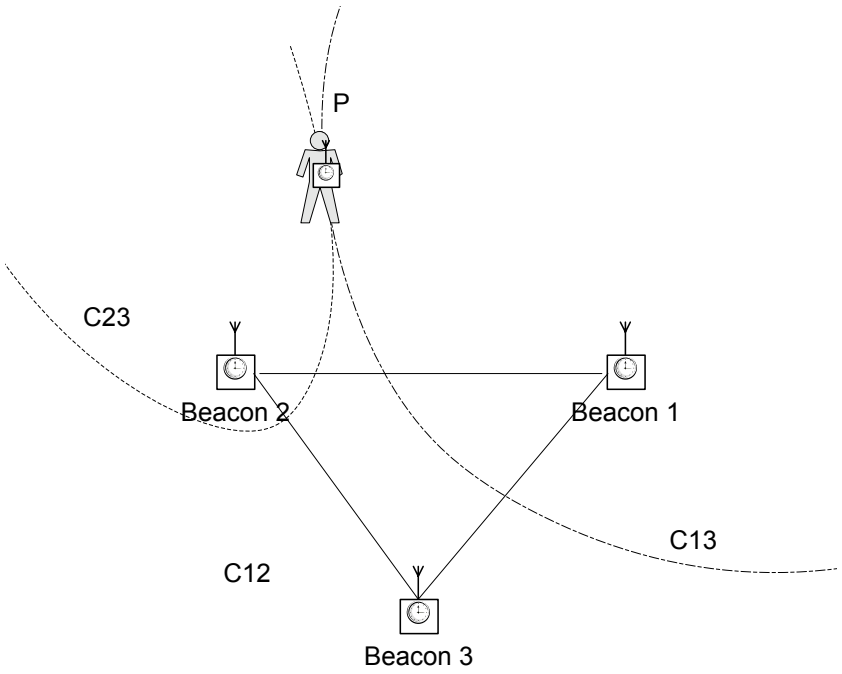


Fig. 4.5: 2D Phase-based TDOA localization with 3 beacons.

The formulation can be converted into the standard TDOA form. After defining d_i as

$$d_i = \frac{(\theta_{iA} - \theta_{1A})\lambda}{n_P} - r_{1i},$$

we have

$$\begin{aligned} d_i &= r_{iA} - r_{1A},\ i \geq 2, \\ z_i &= d_i + e_i,\ i \geq 2. \end{aligned}$$

For simplicity, the notation $r_{jA}, j \geq 1$ is replaced by r_j. That is

$$\begin{aligned} d_i &= r_i - r_1,\ i \geq 2, \\ z_i &= d_i + e_i. \end{aligned} \tag{4.3}$$

The TDOA can be solved by standard non-linear least squares (LS) methods, which can be formulated as the following equations:

$$\begin{aligned} & \min_{\mathbf{p}} J_1(\mathbf{p};\mathbf{q}) \\ J_1 &= \frac{1}{2}\sum_{i=2}^{n} e_i^2 = \frac{1}{2}\sum_{i=2}^{n}(d_i - r_i + r_1)^2, \\ r_i &= \|\mathbf{p} - \mathbf{q}_i\|, \\ \mathbf{q} &= \{q_1, q_2, \cdots, q_n\}, \end{aligned}$$

where $\mathbf{q}_i$ is the position of the i-th beacon; $\mathbf{p}$ is the position of the mobile node; r_i is the distance between mobile node and the i-th beacon; and d_i is defined in (4.3). d_i is subject to measurement noise e_i. The optimal estimation on $\mathbf{p}$ is the $\hat{\mathbf{p}}$ as shown in the following equation:

$$\hat{\mathbf{p}} = \arg\min_{\mathbf{p}} J_1(\mathbf{p};\mathbf{q}). \tag{4.4}$$

4.4 Beacon Placement Optimization

4.4.1 Application Scenarios

From (4.4), it is easy to see that the estimation on $\mathbf{p}$ depends on the positions of the beacons, i.e., $\mathbf{q}$. A natural question to ask is that what is the proper positions for the beacons, in order to guarantee the precision on the estimation of $\mathbf{p}$?

In typical WSN TDOA localization systems, users have the freedom to choose the positions of the beacons. The placement of the beacons may affect the localization error significantly. For example, if any of the two beacons of a 3-beacon TDOA localization system is very close to each other, then one of the beacons is virtually invalid, and the positioning error could be significant. When the domain of interested is unbounded or regular, the placement of the beacons could be intuitive. Beacons should be placed far apart from each other and not align on one line. However, the placement may not be intuitive in practice, for example, when the deployment domain of the beacons is irregular. Imaging a WSN-based localization system is deployed inside a building. The non-restrictive optimal

beacon position could be out of the building, or within concrete walls, or other localizations where beacon placement are impossible. Such constraints should be considered. For outdoor environments, many obstacles, such as lakes, highways, and buildings could pose constraints to the beacon deployments. In addition, as [72] suggested, perturbation factors on beacon placement and spatial noises also call for systematic beacon placement methods.

The beacon placement problem has been discussed in [39, 40, 72] from different aspects. Motivated by improving the precision of proximity localization systems, a heuristic adaptive beacon placement method is presented in [72]. Based on the positioning errors of the current beacon placement in a regular domain, this method select grid points to place beacons to enhance localization precision. In [40] and [39], the beacon placement is optimized for multilateral localization, which are commonly formulated as LS problems. The beacon placement problem is formulated as a binary integer programming problem in [40], where the domain of interested is separated by 16×10 grids. One binary variable is associated with one grid point and indicates whether a beacon is on the point or not. The total number of beacons is minimized by the binary integer programming, subject to a constraint that the maximum distance from beacons to the mobile node must be no more than a certain threshold. The optimal beacon placement pattern is discussed in [39] based on an information theoretical approach. This is also a grid-based algorithm. In an open domain, several beacon placement patterns are compared based on the information entropy of the beacon signals. This method is applicable to domains where there is no constraints, or the domains which are so large that the most of the beacons are placed internally without being affected by the constraints on the boundary of the domain.

The problem studied in this dissertation is close to that in [40] but different. In this dissertation, the beacon placement is optimized for robustness of the positioning. Base on the context of TDOA localization, we firstly solve a direct beacon placement problem, where a given number of beacons are placed within a complex domain such that the maximum positioning error is minimized. In other words, the deployment provides robust, uniform small positioning errors every where in the domain. Next, a progressive beacon

placement problem is addressed. Assuming the estimation precision does not satisfy our requirements and thus the system should be upgraded. Given the existing beacons, the progressive beacon placement method adds a given number of beacons, such that the maximum positioning error based on all the beacons is minimized.

The concept and motivation of the progressive beacon placement are similar to that of the adaptive beacon placement method in [72]. However, irregular domains, robustness and multilateral localization have not been discussed by the latter method.

4.4.2 Problem Formulation

In order to design the beacons' positions, $\mathbf{q}$, properly, we formulate the issue as a rigorous optimization problem and solve it within the framework of optimal experimental design and semi-infinite programming (SIP).

At first, a cost function should be constructed based on the covariance matrix of $\hat{\mathbf{p}}$, or $cov(\hat{\mathbf{p}})$. As what is presented in Chapters 2 and 3, $cov(\hat{\mathbf{p}}) = M^{-1}$, where M is the Fisher information matrix (FIM). Again, the D-optimality criteria, $\Psi(M) = -\ln\det(M)$, is applied to the FIM.

Now, the beacon position optimization problem can be formulated as a min-max problem as follows:

$$\hat{\mathbf{q}} = \arg\min_{\mathbf{q}\in\Omega}\max_{\mathbf{p}\in\Omega}\Psi(M(\mathbf{p};\mathbf{q})), \tag{4.5}$$

$$M = A^T A, \tag{4.6}$$

$$d_i = r_i - r_1,$$

$$r_i = \|\mathbf{p} - \mathbf{q}_i\|,$$

$$A = (\nabla_{\mathbf{p}}\mathbf{d})^T,$$

where Ω is the domain that the beacons and the mobile node may be deployed inside.

To solve the problem in (4.5), the sensitivity, A, is computed as the follows

$$\begin{aligned} A &= (\nabla_{\mathbf{p}}\mathbf{d})^T \\ &= \begin{pmatrix} \frac{\partial}{\partial p_{\cdot x}} d_2 & \cdots & \frac{\partial}{\partial p_{\cdot x}} d_n \\ \frac{\partial}{\partial p_{\cdot y}} d_2 & \cdots & \frac{\partial}{\partial p_{\cdot y}} d_n \end{pmatrix} \\ &= \begin{pmatrix} r_1^{-1}(\mathbf{p}-\mathbf{q}_1) - r_2^{-1}(\mathbf{p}-\mathbf{q}_2) & \cdots & r_1^{-1}(\mathbf{p}-\mathbf{q}_1) - r_n^{-1}(\mathbf{p}-\mathbf{q}_n) \end{pmatrix} \\ &= (\mathbf{a}_2 \cdots \mathbf{a}_n) \end{aligned}$$

$\mathbf{a}_i$ is defined as this

$$\mathbf{a}_i = r_1^{-1}(\mathbf{p}-\mathbf{q}_1) - r_i^{-1}(\mathbf{p}-\mathbf{q}_i).$$

Thus, $\mathbf{a}_i$ is a column of the matrix A.

$$A(:,i) = \mathbf{a}_i,\ i \neq 1.$$

In summary,

$$\begin{aligned} M &= A^T A, \\ &= \sum_{i=2}^{n} \mathbf{a}_i \mathbf{a}_i^T. \end{aligned}$$

This problem can be solved by SIP method. A generic SIP problem is defined as the follows:

Definition 4.4.1 (Semi-infinite Programming) *When $f(x)$ and $g(x,s)$ are two functions, semi-infinite programming is the following optimization problem:*

$$\begin{aligned} \min_x \quad & f(x), \\ \textit{subject to:} \quad & \forall s \in \Omega,\ g(x,s) \leq 0. \end{aligned}$$

Thus, the direction optimal beacon placement problem can be formulated as a SIP problem.

Definition 4.4.2 (Direct Optimal Beacon Placement) *Given acceptable beacon placement domain as Ω_1 and the mobile node's placement domain Ω_2, the direct beacon placement problem is to solve $\mathbf{q}_i,\ i \in [1,n]$ that satisfies the following formulation:*

$$\begin{aligned} \min_{\mathbf{q}_i \in \Omega_1, i \in [1,n]} \quad & \Psi[M(\hat{\mathbf{p}}; \mathbf{q}_i)], \\ \text{subject to:} \quad & \max_{\mathbf{p} \in \Omega_2} \Psi[M(\mathbf{p}; \mathbf{q}_i)] - \Psi[M(\hat{\mathbf{p}}; \mathbf{q}_i)] \leq 0, \\ & \hat{\mathbf{p}} = \arg\max_{\mathbf{p} \in \Omega_2} \Psi[M(\mathbf{p}; \mathbf{q}_i)]. \end{aligned}$$

Remark The equivalence between (4.5) and the definition 4.4.2 is more obvious if the above equations are simplified as:

$$\begin{aligned} \min_{\mathbf{q}} \quad & y(\mathbf{q}), \\ \text{subject to:} \quad & \max_{\mathbf{p}} \Psi(M) \leq y(\mathbf{q}). \end{aligned} \tag{4.7}$$

Although not as precise as Definition 4.4.2, (4.7) captures the key concept and is easy to understand. For simplicities, the examples in this dissertation have the same placement domains for beacons and the mobile node, i.e., $\Omega_1 = \Omega_2$.

The aforementioned progressive optimal beacon placement problem can also be formulated within the SIP framework.

Definition 4.4.3 (Progressive Optimal Beacon Placement) *Given n beacons placed at the positions $\mathbf{q}_i \in \Omega_1\ i \in [1,n]$, the optimal position of additional k beacons are $\mathbf{q}_j \in \Omega_1\ j \in [n+1, n+k]$, which are solved by the following formulation:*

$$\begin{aligned} \min_{\mathbf{q}_j \in \Omega_1, j \in [n+1, n+k]} \quad & \Psi[M(\hat{\mathbf{p}}, \mathbf{q}_j; \mathbf{q}_i)],\ i \in [1,n] \\ \textit{subject to:} \quad & \max_{\mathbf{p} \in \Omega_2} \Psi[M(\mathbf{p}, \mathbf{q}_j; \mathbf{q}_i)] - \Psi[M(\hat{\mathbf{p}}, \mathbf{q}_j; \mathbf{q}_i)] \leq 0, \\ & \hat{\mathbf{p}} = \arg\max_{\mathbf{p} \in \Omega_2} \Psi[M(\hat{\mathbf{p}}, \mathbf{q}_j; \mathbf{q}_i)],\ i \in [1,n]. \end{aligned}$$

4.4.3 Solution and Simulation

All the simulations in this chapter is based on a Matlab function named `fseminf`. The following paragraph is from the help on the function from Matlab.

```
FSEMINF solves problems of the form:
    min { F(x) | C(x)<=0 , Ceq(x) = 0 , PHI(x,w)<=0 }
     x
    for all w in an interval.

    X=FSEMINF(FUN,X0,NTHETA,SEMINFCON) starts at X0 and finds minimum X
    to the function FUN constrained by NTHETA semi-infinite constraints
    in the function SEMINFCON. FUN accepts vector input X and returns
    the scalar function value F evaluated at X.
```

We need to provide two Matlab functions, `Jsip` and `JsipCont`, to characterize the cost function and constraints, which are the `C(x), Ceq(x)` and `PHI(x,w)` in the declaration of `fseminf`. In our simulation, the function is invoked as the following form

```
Q=fseminf(@Jsip,Q0,2,@JsipCont)
```
(4.8)

where `Q0` is the initial guess on `Q`, which is the optimized beacon positions in the following form

$$Q = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n].$$

That is $\mathbf{q}_i = Q(:, i)$. Each $\mathbf{q}_i$ is a column vector that represents the position of a beacon. The key task for the implementation is to incorporate the SIP in the two configuration functions, i.e., `Jsip` and `JsipCont`.

In addition, for convenance, we write a function named `isInDomain` which judges if an array of points are inside the domain of interest or not. The function returns an array after getting a list of positions, Q, as the input. Each entry in the output array is associated with one position in the inputs. If the position of the input is not within the domain of

interest, the associated output entry is -1, otherwise the entry is 1. This output array is actually the `C(x)` constraint in the `fseminf` function. The usage of the `isInDomain` function is shown in Tables 4.1 and 4.2.

The pseudo-code of the direct and progressive optimal beacon placement algorithms are listed in Tables 4.1 and 4.2, respectively. The computation on the FIMs, M_G and M, have been discussed. After the functions `Jsip` and `JsipCont` being called like the form in (4.8), the output is `Q`. Comparing the two algorithms in Tables 4.1 and 4.2, we see that the similarities between the two are significant. The different behaviors between them is mainly due to the fact that the positions for the existing beacons, i.e., Q_S, are a global variable in the `JsipCont` function for the progressive beacon placement algorithm, but not in the direct beacon placement algorithm. The positions for the beacons yet to be optimized is embedded in Q, which is the input variable for `Jsip` and `JsipCont` functions of the two algorithms. In other words, Q is configured as the parameter to be optimized, but Q_S is not.

The input variable s for the two `JsipCont` functions in Tables 4.1 and 4.2 are not used, since s is required for internal usage by the Matlab optimization process only. At this point, it is easy to understand the number 2 in (4.8), which is the number of semi-infinite constraints. Observing from the `JsipCont` functions in Tables 4.1 and 4.2 that there are two semi-infinite constraints, i.e., `PhiCon` and `QCon`, in the returning parameter list for each `JsipCont` function. The global variable $\mathbf{p}$ is the max error position of the mobile node, where the associated D_M entry has the largest value. Thus, $\mathbf{p}$ is the pseudo-code is $\arg\max_{\mathbf{p}\in\Omega} \Psi(M)$.

Observe the pseudo-code, it is easy to see that other multilateral localization methods, such as TOA and AOA, can be studied in the same framework also. Almost everything in the pseudo-code is applicable, except the computations on FIMs are different. Thus, the proposed beacon placement methods are not limited to TDOA localization only.

The simulation results are included in Figs. 4.6 to 4.11. In Figs. 4.6 to 4.10, the red diamond shapes indicate the optimized beacon positions. If the beacons are placed on

Table 4.1: Pseudo-Code for the Direct Optimal Beacon Placement

```
function J = Jsip(Q)
global p M_G;

M_G = sum_{i=1}^{n} a_i(p;q_i)a_i(p;q_i)^T;
J=Psi(M_G);
return;

function [c,ceq,PhiCon,QCon,s] = JsipCont(Q,s)
global p M_G;
% Q is the positions of all the beacons. Q = [q_1, q_2, ..., q_n].
p=meshgrid(linspace(0,1,n_G),linspace(0,1,n_G)); % segment the domain
    that surround the Omega into n_G x n_G grids.
For each grid point p(r,c), where r and c are the row and column
    indices, respectively.
    If p(r,c) in Omega
        D_M(r,c) = Psi{M[p(r,c);Q]};
    else
        D_M(r,c) = 0;
    end
end
PhiCon=D_M - Psi(M_G)*ones(size(D_M));
QCon=isInDomain(Q);
p=the position associated with max D_M;
c=[];
ceq=[];
return;
```

the positions of the red diamonds, the worst case position estimation in this domain of minimized. Based on those figures and our experiments, we have the following observations

- There is virtually no restriction on the shape of the domain of interest. As far as the function `isInDomain` distinguish the internal and external positions of the domain, the algorithms in Tables. 4.1 and 4.2 are valid. As a comparison, the algorithms in [39, 72] are not applicable for arbitrary domains.

- Also, there is no restriction on the number of beacons to be placed or have been placed. For example, in Fig. 4.11, 5 beacons have been placed, and the positions of the 2 additional beacons are optimized. In Figs. 4.6 to 4.9, the deployment on 3

Table 4.2: Pseudo-Code for the Progressive Optimal Beacon Placement

```
function J = Jsip(Q)
global p M_G Q_S; % Q_S are the static beacons which have been placed.
% Q_S = [q_S1, q_S2, ..., q_Sn].
% Q is the positions of the additional beacons. Q = [q_{n+1}, q_{n+2}, ..., q_{n+k}].

M_G = sum_{i=1}^{n} a_i(p; q_Si) a_i(p; q_Si)^T + sum_{j=n+1}^{n+k} a_j(p; q_j) a_j(p; q_j)^T;
J=Psi(M_G);
return;

function [c, ceq, PhiCon, QCon, s] = JsipCont(Q, s)
global p M_G Q_S;

p=meshgrid(linspace(0,1,n_G), linspace(0,1,n_G)); % segment the domain
    that surround the Omega into n_G x n_G grids.
For each grid point p(r,c), where r and c are the row and column
    indices, respectively.
    If p(r,c) in Omega
        D_M(r,c) = Psi{M[p(r,c); Q_S]} + Psi{M[p(r,c); Q]};
    else
        D_M(r,c) = 0;
    end
end
PhiCon=D_M - Psi(M_G)*ones(size(D_M));
QCon=isInDomain(Q);
p=the position associated with max D_M;
c=[];
ceq=[];
return;
```

beacons are optimized. It can be seen from the pseudo-code in Tables. 4.1 and 4.2 that beacon number does not have restrictions.

- The speed of the simulation is fast. On a Pentium 4 3Ghz PC, the execution time is from about 9 sec. to 160 sec. Considering that the grid size in our simulation is $100 \times 100 = 10,000$ ($n_G = 100$ in Tables. 4.1 and 4.2), which is considerable large, the speed of the simulation is fast.

- The optimal beacon placement for irregular domains may not be intuitive. For example, intuitively, the right beacon in Fig. 4.8 may be placed on the right-most

hemisphere boundary of the domain, since this position ensure that more area are covered by the triangle make up from the three beacons. However, the result of the optimization does not support this intuition. In fact, the beacon placements in Figs. 4.7 to 4.11 are not very intuitive also. These result supports the claim that systematic optimization on the beacon placement is necessary.

- We observed that the optimized solution may not be unique. For example, the domain of interest in Fig. 4.6 is a square, which is symmetric. The optimal positions of the beacons must be symmetric. Thus, if the beacon positions in Fig. 4.6 are rotated by 90 degrees, the new positions should have the same cost as the current solution.

- Follow the above observation, it seems that, similar to common non-linear optimization, the outputs of the algorithms are local optimum solutions. The global minimum solution is not guaranteed to be found. In the simulations, we observed that the initial values affect the final results. In practice, it is suggested to execute the optimization several times with different random initial values. This simple strategy helps to find the global minimal solution. Of course, there are many other strategies for the global optimization, which is a rich research topic. Due to limited space, no further discussion on the global optimization is presented.

4.5 Chapter Summary

In this chapter, an asynchronous TDOA localization method for energy efficient and low cost WSN localization is proposed, namely, phase of arrival localization. The solution can be transferred into the standard TDOA formulation. In order to minimize the maximum positioning errors within deployment domains, two algorithms are proposed to optimize the placement of the beacons, based on SIP (semi-infinite programming) approaches. These two algorithms are targeted at direct and progressive optimal beacon placement, respectively. Comparing to the related beacon placement methods, the proposed approaches are fast and have no constraints the shapes of the beacon deployment domains.

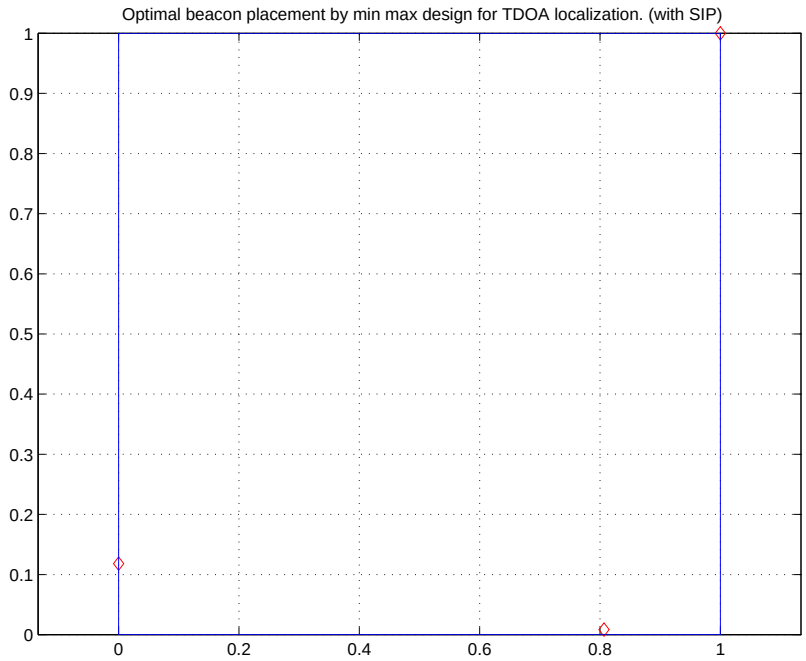


Fig. 4.6: Direct optimal beacon placement for TDOA localization, domain 1.

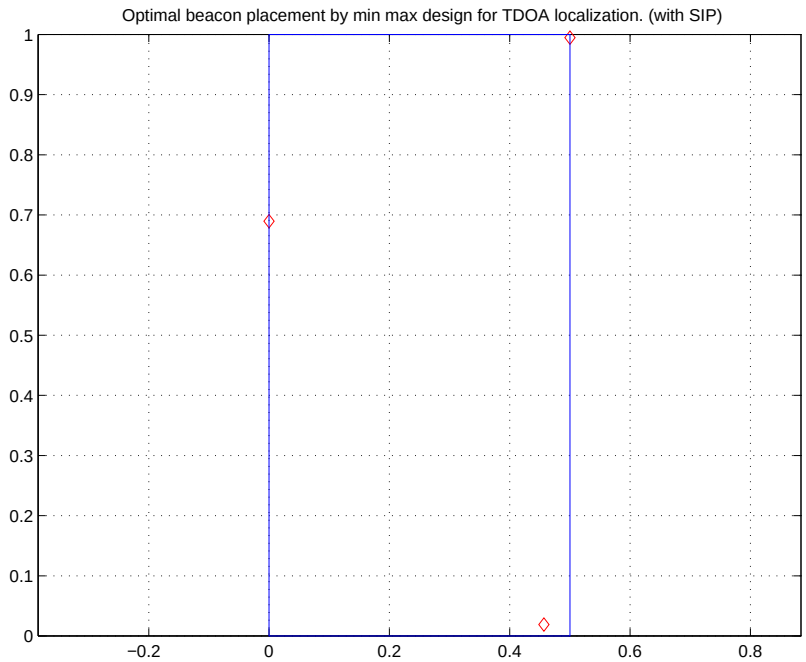


Fig. 4.7: Direct optimal beacon placement for TDOA localization, domain 2.

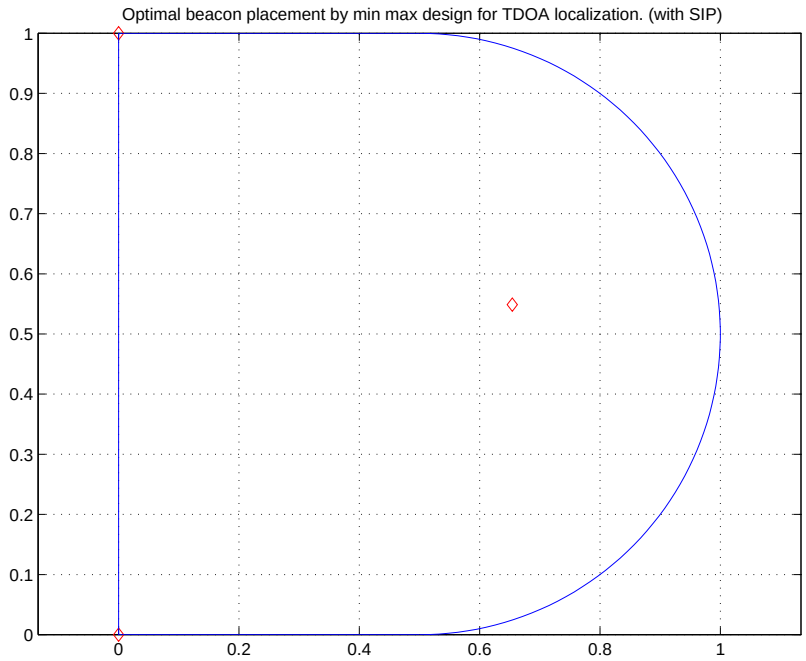


Fig. 4.8: Direct optimal beacon placement for TDOA localization, domain 3.

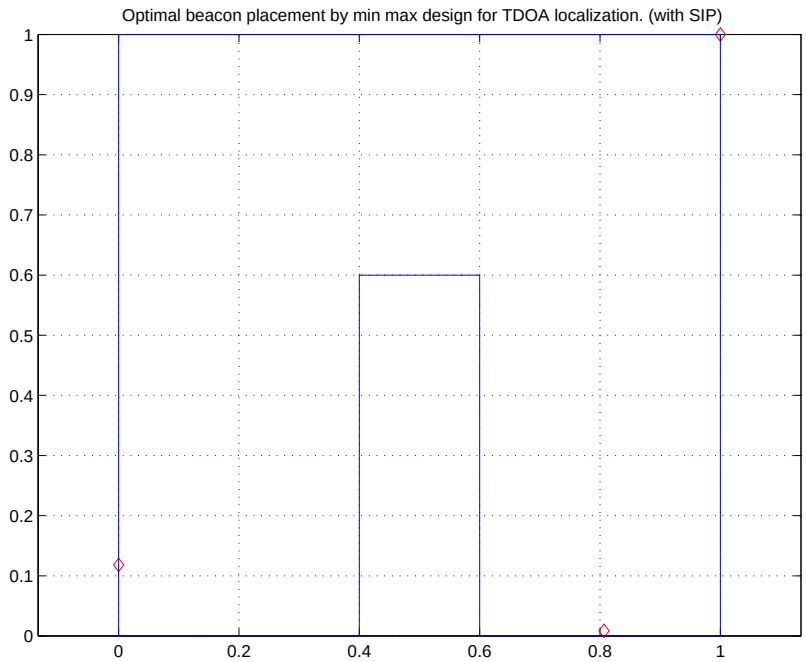


Fig. 4.9: Direct optimal beacon placement for TDOA localization, domain 4.

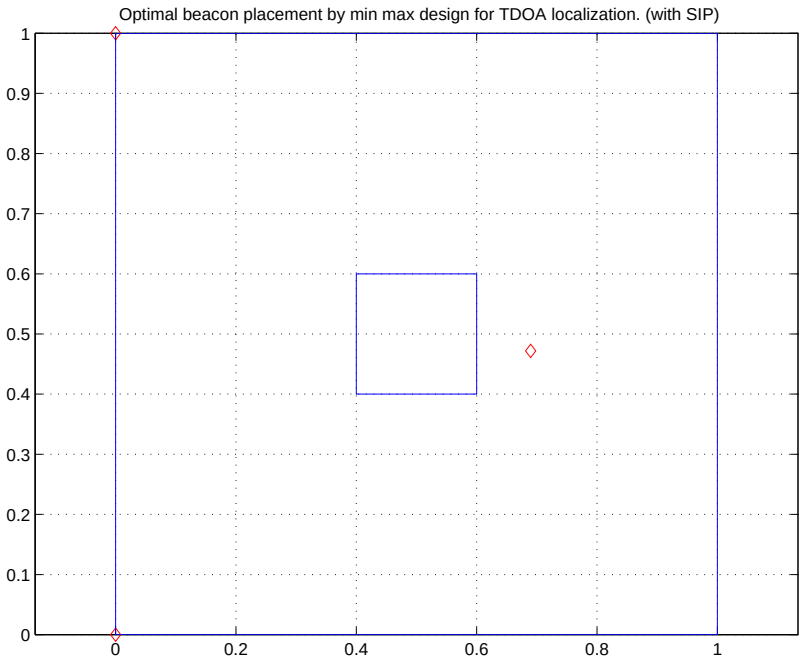


Fig. 4.10: Direct optimal beacon placement for TDOA localization, domain 5.

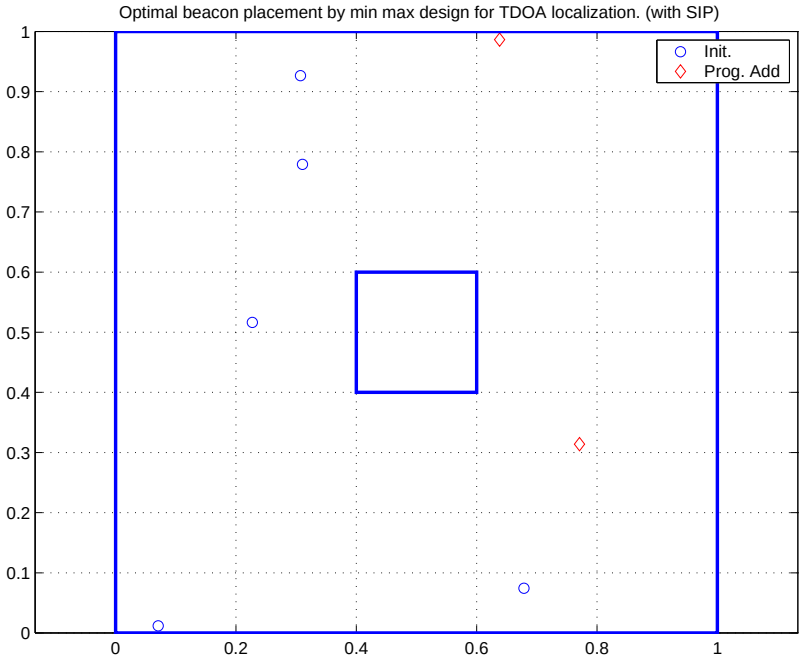


Fig. 4.11: Progressive optimal beacon placement for TDOA localization, domain 5.

Chapter 5
Conclusion

Wireless sensor network (WSN) is an important technology for physical world observation. While WSN has significant advantages over the traditional wired data collection systems, it is facing unique challenges. For example, the massive and densely deployed WSN could provide more cost-efficient and fine-grid sensing on physical phenomena, comparing to the traditional methods. The mobilities of the wireless sensor nodes may further enhance the performance on the observation. To facilitate the above applications, many challenges must be addressed, including energy efficient observation, localization, scalability, fault tolerance, etc. Among them, energy efficiency is a central challenge that has caught much attentions and been attacked from different aspects, but is still under active research. Much of this dissertation is motivated by reducing energy costs. This discussion discusses on several key challenges of WSNs within a unified theoretical framework, i.e, the theory of optimal experimental design.

Our first challenge is to take the mobilities of sensors for better observation on physical phenomena. In order to drive mobile sensors along the optimum trajectories for parameter estimations on distribution parameter systems (DPSs), a numerical optimal control method is proposed.

The second challenge is to design energy efficient observation methods. A sensor selection algorithm named COSS (Convex Optimal Sensor Selection) is designed and analyzed. We prove that there is a class of Implicit Optimal Sensor Selection (IOSS) methods, which includes the COSS algorithm and can achieve the optimal estimation predicted by the CRLB (Cramér-Rao Lower Bound) with the minimal number of sensors that allowed by the Carathéodory's theorem. The robustness and efficiency of the COSS are tested extensively by simulation and hardware experiments.

The last but not the least important challenge is WSN localization. Motivated by reducing hardware and energy costs, an asynchronous TDOA (Time Difference of Arrival) method is proposed, namely, phase of arrival method, which can be converted into the standard TDOA problem formulation. Based on the TDOA method, two fast beacon placement optimization algorithms are developed based on SIP (semi-infinite programming). The methods are applicable not only to the proposed asynchronous TDOA, but also generic TDOA and TOA (Time of Arrival) as well. Comparing to the related beacon placement methods, the proposed approaches are fast and have no constraints the shapes of the beacon deployment domains.

All those challenges are formulated as convex optimizations on cost functions based on the FIM (Fisher information matrix), which plays a key role in the theory of optimal experimental design. The abstractions of the three problems in this dissertation share the same theoretical framework, and highly related to each other.

References

[1] I. Oppermann, M. Jamtgaard, L. Ouvry, and P. Rouzet, "Aetherwire 15.4a CFP response" [Online]. Available: http://www.ieee802.org/15/pub/05/.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[3] C.-Y. Chong and S. Pumar, "Sensor networks: Evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, Aug. 2003.

[4] National Instruments, "Labview: 20 years of innovation," Jan. 2007 [Online]. Available: http://www.ni.com/labview/.

[5] F. Zhao and L. Guibas, *Wireless Sensor Networks: An Information Processing Approach.* Morgan Kaufmann Publishers, 2004.

[6] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2001.

[7] H. Qi, S. S. Iyengar, and K. Chakrabarty, "Distributed sensor networks – a review of recent research," *Journal of the Franklin Institute*, vol. 338, pp. 655–668, 2001.

[8] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 2033–2036, May 2001.

[9] B. Krishnamachari, D. Estrin, and S. Wicker, "Modelling data-centric routing in wireless sensor networks," 2002 [Online]. Available: http://citeseer.ist.psu.edu/article/krishnamachari02modelling.html.

[10] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis, "Interference-aware fair rate control in wireless sensor networks," in *Proceedings of ACM SIGCOMM Symposium on Network Architectures and Protocols*, Pisa, Italy, Sep. 2006.

[11] B. Warneke, M. Last, B. Liebowitz, and K. Pister, "Smart dust: communicating with a cubic-millimeter computer," *Computer*, vol. 34, no. 1, pp. 44–51, Jan. 2001.

[12] IEEE Computer Society, "Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)," Oct. 2003 [Online]. Available: http://standards.ieee.org/catalog/olis/lanman.html.

[13] K. Lorincz and M. Welsh, "MoteTrack: A robust, decentralized approach to RF-based location tracking," *Springer Personal and Ubiquitous Computing, Special Issue on Location and Context-Awareness*, pp. 1617–4909, Oct. 2006 [Online]. Available: http://www.eecs.harvard.edu/~konrad/projects/motetrack.

[14] M. C. Vuran, O. B. Akan, and I. F. Akyildiz, "Spatio-temporal correlation: Theory and applications for wireless sensor networks," *Computer Networks Journal (Elsevier)*, vol. 45, no. 3, pp. 245 –259, June 2004.

[15] Wikipedia, "Piconet — wikipedia, the free encyclopedia," 2006, accessed 8-February-2007 [Online]. Available: http://en.wikipedia.org/w/index.php?title=Piconet&oldid=93824496.

[16] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 364–369, Apr. 2005 [Online]. Available: http://www.moteiv.com/products/docs/an002-telos.pdf.

[17] J. Gehrke and S. Madden, "Query processing in sensor networks," *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 46–55, Jan.-Mar. 2004.

[18] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, "CodeBlue: An ad-hoc sensor network infrastructure for emergency medical care" [Online]. Available: http://www.eecs.harvard.edu/~mdw/papers/codeblue-bsn04.pdf.

[19] P. Ögren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1292–1302, Aug. 2004.

[20] P. K. Dutta, A. K. Arora, and S. B. Bibyk, "Towards radar-enabled sensor networks," in *In The Fifth International Conference on Information Processing in Sensor Networks (IPSN'06) Special track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS '06)*, pp. 467–474, Apr. 2006.

[21] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Computer Networks, Special Issue on Military Communications Systems and Technologies*, vol. 46, no. 5, pp. 605–634, July 2004.

[22] Crossbow, "Life fitness exercises efficiencies with wireless mesh networking," 2006 [Online]. Available: http://www.xbow.com/Industry_solutions/BuildingAutomation.aspx.

[23] T. Kevan, "Shipboard machine monitoring for predictive maintenance," *Sensors Magazine*, Feb. 2006 [Online]. Available: http://www.sensorsmag.com/sensors/article/articleDetail.jsp?id=314716.

[24] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based counter sniper system," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, Maryland, USA, Nov. 2004.

[25] J. Liu, P. Cheung, L. Guiba, and F. Zhao, "A dual-space approach to tracking and sensor management in wireless sensor networks," Palo Alto Research Center, Tech. Rep. P2002-10077, Mar. 2002.

[26] D. M. Doolina and N. Sitara, "Wireless sensors for wildfire monitoring," in *Proceedings of SPIE Symposium on Smart Structures & Materials*, 2005 [Online]. Available: http://firebug.sourceforge.net/publications.php.

[27] P. K. Dutta and A. K. Arora, "Integrating micropower impulse radar and motes," Ohio State University, Tech. Rep. OSU-CISRC-12/03-TR67, 2004 [Online]. Available: http://www.tinyos.net/dist-1.1.0/snapshot-1.1.5Mar2004cvs/contrib/OSU/ALineInTheSand/MIRSensor/mir.pdf.

[28] D. E. Culler, D. Estrin, and M. B. Srivastava, "Guest editors' introduction: Overview of sensor networks." *IEEE Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[29] H.-J. Krber, H. Wattar, G. Scholl, and W. Heller, "Embedding a microchip PIC18F452 based commercial platform into TinyOS," in *Workshop on Real-World Wireless Sensor Networks (REALWSN)*, 2005 [Online]. Available: http://www.sics.se/realwsn05/.

[30] Sensicast Inc., "Sensicast EMS for environmental management" [Online]. Available: http://www.sensicast.com/downloads/brochure_controls_ems.pdf.

[31] Nanotron Inc., "nanoNET chirp-based wireless networks," 2004, ver. 1.02 [Online]. Available: http://www.nanotron.com/nanotron_intern/en/technology/css.php?navId=20.

[32] J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, and S. Roundy, "Picoradio supports ad hoc ultra-low power wireless networking," *Computer*, vol. 33, no. 7, pp. 42–48, Jul. 2000.

[33] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, New York, NY, USA, Jun. 2002 [Online]. Available: http://www.isi.edu/scadds/projects/smac/.

[34] TinyOS group, "Multihop routing," Sept. 2003 [Online]. Available: http://www.tinyos.net/tinyos-1.x/doc/multihop/multihop_routing.html.

[35] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao, "Towards a sensor network architecture: lowering the waistline," in *Proceedings of the Tenth Workshop on Hot Topics in Operating Systems (HotOS X)*, 2005 [Online]. Available: http://www.cs.berkeley.edu/~pal/pubs/sna.pdf.

[36] J. Feng, G. Qu, and M. Potkonjak, "Actuator-based infield sensor calibration," *IEEE Sensors Journal*, vol. 6, no. 6, pp. 1571–1579, 2006.

[37] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks," in *Workshop on Wireless Sensor Networks and Applications*, 2005.

[38] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich, "Collaborative signal and information processing: An information-directed approach," *IEEE Proceedings*, vol. 91, no. 8, pp. 1199–1209, Aug. 2003.

[39] H. Wang, K. Yao, and D. Estrin, "Information-theoretic approaches for sensor selection and placement in sensor networks for target localization and tracking : Communication and signal processing in sensor networks," *Journal of communication and networks*, vol. 7, no. 4, pp. 438–449, 2005.

[40] J. Yick, A. Bharathidasan, G. Pasternack, B. Mukherjee, and D. Ghosal, "Optimizing placement of beacons and data loggers in a sensor network - a case study," in *IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 4, pp. 2486–2491 Vol.4, 2004 [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1311479.

[41] Z. Song, Y. Chen, J. Liang, and D. Uciński, "Optimal mobile sensor motion planning under nonholonomic constraints for parameter estimation of distributed parameter systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3163–3168, Aug. 2005.

[42] H. Cao, E. Ertin, V. Kulathumani, M. Sridharan, and A. Arora, "Differential games in large-scale sensor-actuator networks." in *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 77–84, Nashville, Tennessee, Apr. 2006 [Online]. Available: http://doi.acm.org/10.1145/1127777.1127792.

[43] A. Meliou, D. Chu, J. M. Hellerstein, C. Guestrin, and W. Hong, "Data gathering tours in sensor networks," in *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN)*. Nashville, Tennessee: ACM, Apr. 2006.

[44] T. H. Chung, V. Gupta, J. W. Burdick, and R. M. Murray, "On a decentralized active sensing strategy using mobile sensor platforms in a network," in *43rd IEEE Conference on Decision and Control (CDC)*, Atlantis, Paradise Island, Bahamas, Dec. 2004.

[45] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *Third International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 1–10, Apr. 26-27 2004 [Online]. Available: http://doi.acm.org/10.1145/984622.984624.

[46] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 168–176, Nashville, TN, Apr. 2006.

[47] L. A. Rossi, B. Krishnamachari, and C.-C. J. Kuo, "Distributed parameter estimation for monitoring diffusion phenomena using physical models," in *IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2nd, Ed., pp. 460–469. Santa Clara, CA.: Kluwer Academic/Plenum Publishers, Oct. 4-7 2004.

[48] A. S. Chhetri, D. Morrell, and A. Papandreou-Suppappola, "Sensor scheduling using a 0-1 mixed integer programming framework," in *IEEE Workshop on Sensor Array and Multichannel Processing*, pp. 471–475, Waltham, Massachusetts, July 2006.

[49] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 61–72, Mar. 2002.

[50] J. Aslam, Z. Butler, F. C. andValentino Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," in *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys)*, pp. 150–161. Los Angeles, California, USA: ACM Press, New York, NY, USA, 2003.

[51] S. Aranda, S. Martinez, and F. Bullo, "On optimal sensor placement and motion coordination for target tracking," in *International Conference on Robotics and Automation*, Barcelona, Spain, Apr. 2005.

[52] D. Uciński, *Optimal measurement methods for distributed parameter system identification*. Boca Raton, Florida: CRC press, 2005.

[53] V. Fedorov and P. Hackl, "Optimal experimental design: Spatial sampling," *Calcutta Statistical Association Bulletin*, vol. 44, pp. 173–174, 1994 [Online]. Available: http://citeseer.ist.psu.edu/fedorov94optimal.html.

[54] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *Proceedings of ACM Sigmod*, pp. 491–502. ACM Press, 2003.

[55] K. L. Moore and Y. Q. Chen, "Model-based approach to characterization of diffusion processes via distributed control of actuated sensor networks," in *The 1st IFAC Symposium on Telematics Applications in Automation and Robotics*. Helsinki University of Technology Espoo, Finland: IFAC, Jun. 2004.

[56] Y. Chen, K. L. Moore, and Z. Song, "Diffusion boundary determination and zone control via mobile actuator-sensor networks (MAS-net): Challenges and opportunities," in *Intelligent Computing: Theory and Applications II, part of SPIE's Defense and Security*. Orlando, FL, USA: SPIE, Apr. 2004.

[57] K. L. Moore, Y. Chen, and Z. Song, "Diffusion based path planning in mobile actuator-sensor networks (MAS-net) - some preliminary results," in *Proc. of SPIE Conf. on Intelligent Computing: Theory and Applications II, part of SPIE's Defense and Security*. Orlando, FL, USA: SPIE, Apr. 2004.

[58] Z. Wang, Z. Song, P.-Y. Chen, A. Arora, K. L. Moore, and Y. Chen, "MASmote–a mobility node for MAS-net (mobile actuator sensor networks)," in *IEEE International Conference on Robotics and Biomimetics*, pp. 816–821. Shenyang, China: IEEE, Aug. 2004.

[59] P. Chen, "Pattern formation in mobile wireless sensor networks," Master's thesis, Utah State University, Logan, Utah, USA, 2005.

[60] A. Arora, "Sensing a diffusion process using distributed robots," Master's thesis, Utah State University, Logan, Utah, USA, 2005.

[61] K. L. Moore, "Coordination and control of distributed networks of actuated sensors – a white paper," Center for Self-Organizing and Intelligent Systems (CSOIS), Utah State University, Logan, UT, Tech. Rep., Mar. 2003 [Online]. Available: http://www.engineering.usu.edu/ece/faculty/moorek/mooreweb.html.

[62] P. Chen and Z. Song, "Cooperative fog estimation by masmote robots." http://mechatronics.ece.usu.edu/mas-net/index_files/Movie_FogDemo.avi, Feb. 2005, also available at http://mechatronics.ece.usu.edu/mas-net/index_files/Page478.htm "Fog Detection" section.

[63] D. Niculescu and B. Nath, "Localized positioning in ad hoc networks," *Elsevier journal of Ad Hoc Networks*, vol. 1, no. 2-3, pp. 247–259, Sept. 2003 [Online]. Available: http://paul.rutgers.edu/~dnicules/research/.

[64] J. Borenstein, H. R. Everett, and L. Feng, "Where am I, sensors and methods for mobile robot positioning," 1996 [Online]. Available: http://www-personal.umich.edu/~johannb/shared/pos96rep.pdf.

[65] S. Thrun, "Probabilistic algorithms in robotics," Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-00-126, Apr. 2000 [Online]. Available: http://www.ri.cmu.edu/pubs/pub_3353.html.

[66] K. W. Cheung, H. C. So, W. K. Ma, and Y. T. Chan, "Least squares algorithms for time-of-arrival-based mobile location," *IEEE Transactions on Signal Processing*, vol. 52, no. 4, pp. 1121–1130, 2004 [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1275684.

[67] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AOA," in *IEEE Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 3, pp. 1734–1743, 2003 [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1209196.

[68] N. Patwari, J. Ash, S. Kyperountas, R. Moses, N. Correal, and A. Hero, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54– 69, Jul. 2005.

[69] Aether Wire & Locations, Inc., "Ultra-wideband localizers: Executive summary," 1999 [Online]. Available: http://www.aetherwire.com/Aether_Wire/aether.html.

[70] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Trans. Sensor Network*, vol. 2, no. 2, pp. 188–220, 2006.

[71] D. Niculescu, *Forwarding and Positioning Problems in Ad Hoc Networks*. Ph.D. dissertation, Rutgers, The State University of New Jersey, 2004 [Online]. Available: http://paul.rutgers.edu/~dnicules/research/.

[72] N. Bulusu, J. Heidemann, and D. Estrin, "Adaptive beacon placement," in *ICDCS '01: Proceedings of the The 21st International Conference on Distributed Computing Systems*, p. 489. Washington, DC, USA: IEEE Computer Society, 2001.

[73] A. Anandarajah, K. Moore, A. Terzis, and I. J. Wang, "Sensor networks for landslide detection," in *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys)*, pp. 268–269. New York, NY, USA: ACM Press, 2005 [Online]. Available: http://dx.doi.org/10.1145/1098918.1098948.

[74] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing, Special Issue on Data-Driven Applications in Sensor Networks*, vol. 10, no. 2, pp. 18–25, Mar.-Apr. 2006.

[75] D. Zarzhitsky, D. F. Spears, D. R. Thayer, and W. M. Spears, "Agent-based chemical plume tracing using fluid dynamics," *Lecture Notes in omputer Science, Springer-Verlag*, vol. 3228, pp. 146–160, 2004 [Online]. Available: http://www.cs.uwyo.edu/~dspears/cpt/.

[76] P. Christofides, *Nonlinear and Robust Control of PDE Systems: Methods and Applications to Transport-Reaction Processes, Foundations and Applications*, ser. Systems and Control. Boston: Birkhäuser, 2001.

[77] R. Curtain and H. Zwart, *An Introduction to Infinite-Dimensional Linear Systems Theory, Texts in Applied Mathematics*. New York: Springer-Verlag, 1995.

[78] I. Lasiecka and R. Triggiani, *Control theory for partial differential equations: continuous and approximation Theories*, ser. Encyclopedia of Mathematics and Its Applications, Vol. I and II. Cambridge: Cambridge University Press, 2000.

[79] P. Neittaanmäki and D. Tiba, *Optimal Control of Nonlinear Parabolic Systems*, ser. Algorithms, and Applications, Monographs and Textbooks in Pure and Applied Mathematics. New York: Marcel Dekker, Inc., 1994.

[80] S. Omatu and J. H. Seinfeld, *Distributed Parameter Systems: Theory and Applications*, ser. Oxford Mathematical Monographs, O. U. Press, Ed., New York, 1989.

[81] H. Zwart and J. Bontsema, "An application-driven guide through infinite-dimensional systems theory," in *European Control Conference*, G. Bastin and M. Gevers, Eds., pp. 289–328, CIACO, Ottignies/Louvain-la-Neuve, 1997, plenaries and Mini-Courses.

[82] C. S. Kubrusly and H. Malebranche, "Sensors and controllers location in distributed systems: a survey," *Automatica*, vol. 21, no. 2, pp. 117–128, 1985.

[83] D. Uciński, "Optimal sensor location for parameter estimation of distributed processes," *International Journal of Control*, vol. 73, no. 13, pp. 1235–1248, 2000.

[84] E. Rafajłowicz, "Optimum choice of moving sensor trajectories for distributed parameter," *International Journal of Control*, vol. 43, no. 5, pp. 1441–1451, 1986.

[85] D. Uciński and Y. Chen, "Time optimal path planning of moving sensors for parameter estimation of distributed systems," in *44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pp. 5257–5262, 2005.

[86] L. Carotenuto, P. Muraca, and G. Raiconi, "Optimal location of a moving sensor for the estimation of a distributed-parameter system," *International Journal of Control*, vol. 46, no. 5, pp. 1671–1688, 1987.

[87] A. Y. Khapalov, "Optimal measurement trajectories for distributed parameter systems," *Systems and Control Letters*, vol. 18, no. 6, pp. 467–477, 1992.

[88] K. Nakano and S. Sagara, "Optimal measurement problem for a stochastic distributed parameter system with movable sensors," *International Journal of Systems Science*, vol. 12, no. 12, pp. 1429–1445, 1981.

[89] K. Nakano and S. Sagara, "Optimal scanning measurement problem for a stochastic distributed-parameter system," *International Journal of Systems Science*, vol. 19, no. 7, pp. 1069–1083, 1988.

[90] S. Martínez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, pp. 661–668, 2006.

[91] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. Sastry, "Distributed control applications within sensor networks," *IEEE Proceedings*, vol. 91, no. 8, pp. 1235–1249, Aug. 2003.

[92] M. Patan, *Optimal observation strategies for parameter estimation of distributed systems*. Ph.D. dissertation, University of Zielona Gòra, Zielona Gòra, 2004.

[93] R. K. Mehra, "Optimization of measurement schedules and sensor designs for linear dynamic systems," *IEEE Transaction on Automatic Control*, vol. 21, p. 1, 1976.

[94] W. G. Müller, *Collecting spatial data. Optimum design of experiments for random fields*, 2nd ed., ser. Contributions to Statistics. Heidelberg: Physica-Verlag, 2001.

[95] E. Rafajlowicz, "Design of experiments for parameter identification of the static distributged system," *Systems Science*, vol. 4, no. 4, pp. 349–361, 1978.

[96] A. El Jaï and A. J. Pritchard, *Sensors and Actuators in Distributed Systems Analysis*, ser. Ellis Horwood Series in Applied Mathematics. Chichester, West Sussex: Ellis Horwood: Ellis Horwood, John Wiley, 1988.

[97] Mathworks, "Partial differential equation toolbox," 2005 [Online]. Available: http://www.mathworks.com/products/pde/.

[98] Y. Chen and A. L. Schwartz, *Recent developments in optimization and optimal control in chemical engineering*, ch. RIOTS_95 – a MATLAB toolbox for solving general optimal control problems and its applications to chemical processes. Rein Luus Editor, Transworld Research Publishers, 2002 [Online]. Available: http://www.csois.usu.edu/ilc/riots/.

[99] A. Schwartz, E. Polak, and Y. Chen, "RIOTS a matlab toolbox for solving optimal control problems," May 1997 [Online]. Available: http://www.schwartz-home.com/~adam/RIOTS/.

[100] V. Isler and R. Bajcsy, "The sensor selection problem for bounded uncertainty sensing models," *IEEE Transaction on Automation Science and Engineering*, vol. 3, no. 4, pp. 372–381, Oct. 2006, accepted [Online]. Available: http://www.cs.rpi.edu/~isler/new/pub/pubs/isler05tase.pdf.

[101] D. Uciński and M. Patan, "D-optimal design of a monitoring network for parameter estimation of distributed systems," *journal of global optimization*, 2006, accepted.

[102] J. Byers and G. Nasser, "Utility-based decision-making in wireless sensor networks," in *International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 143–144, 2000.

[103] F. Bian, D. Kempe, and R. Govindan, "Utility-based sensor selection," in *International Conference on Information Processing in Sensor Networks (IPSN)*, Nashville, Tennessee, USA, Apr. 2006 [Online]. Available: http://citeseer.ist.psu.edu/747271.html.

[104] J. Liu, J. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," *EURASIP Journal on Applied Signal Processing*, vol. 4, pp. 378–391, Mar. 2003 [Online]. Available: http://citeseer.ist.psu.edu/liu02collaborative.html.

[105] H. Wang, K. Yao, G. Pottie, and D. Estrin, "Entropy-based sensor selection heuristic for target localization," in *Proceedings of the Third International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 36–45. ACM Press, 2004 [Online]. Available: http://citeseer.ist.psu.edu/wang04entropybased.html.

[106] Moteiv Corporation, "Tmote sky datasheet," Nov. 2006 [Online]. Available: http://www.moteiv.com/products/tmotesky.php.

[107] A. S. Chhetri, D. Morrell, and A. Papandreou-Suppappola, "Energy efficient target tracking in a sensor network using non-myopic sensor scheduling," in *IEEE International Conference on Information Fusion*, Philadelphia, Pennsylvania, 2005.

[108] A. L. Buczak, H. H. Wang, H. Darabi, and M. A. Jafari, "Genetic algorithm coverengce study for sensor network optimization," *Information Sciences*, vol. 133, pp. 267–282, 2001.

[109] A. F. Emery and A. V. Nenarokomovz, "Optimal experiment design," *Measurement science & technology*, vol. 9, no. 6, pp. 864–876, 1998 [Online]. Available: http://www.iop.org/EJ/abstract/0957-0233/9/6/003.

[110] A. Pazman, *Foundations of optimum experimental design*, A. Bialynicki-Birula et. al., Ed. D. Reidel Publishing Company, 1986.

[111] S. D. Silvey, *Optimal design. An introduction to the theory for parameter estimation.* Chapman and Hall, 1980.

[112] D. Bertsekas, *Nonlinear Programming.* Athena Scientific, 1999.

[113] Z. Song, "Experiment results of the convex optimal sensor selection (COSS) method," http://cc.usu.edu/~zhensong/cdc07/, Feb. 2007, experiments are due to the collaborations of Zhen Song and Nazif Cihan Tas.

[114] G. C. Goodwin and R. L. Payne, *Dynamic system identification: experiment design and data analysis.* New York: Academic, 1977.

[115] D. Uciński, "Sensor network design for parameter estimation of distributed systems using nonsmooth optimality criteria," in *44th IEEE Conference on Decision and Control, and the European Control Conference*, 2005.

[116] E. Ertin, J. W. Fisher, and L. C. Potter, "Maximum mutual information principle for dynamic sensor query problems," in *Procedings of International Conference on Information Processing in Sensor Networks (IPSN)*, Palo Alto, CA, Apr. 2003.

[117] A. S. Chhetri, *Sensor scheduling and efficient algorithm implementation for target tracking.* Ph.D. dissertation, Arizona State University, 2006.

[118] R. Olfati-Saber, "Distributed tracking for mobile sensor networks with information-driven mobility," in *IEEE Conference on Decision and Control*, 2006.

[119] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Elsevier Computer Networks*, vol. 47, no. 7, pp. 445–487, 2004 [Online]. Available: http://www.ece.gatech.edu/research/labs/bwn/mesh.pdf.

[120] N. B. Priyantha, *The Cricket Indoor Location System.* Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[121] Chipcon, *CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*, 2005 [Online]. Available: http://www.chipcon.com/files/CC2420_Data_Sheet_1_4.pdf.

[122] D. Niculescu, "Positioning in ad hoc sensor networks," *IEEE Network Magazine*, pp. 24–29, July/Aug. 2004.

[123] D. Lymberopoulos, Q. Lindsey, and A. Savvides, "An empirical analysis of radio signal strength variability in ieee 802.15.4 networks using monopole antennas," in *In proceedings of European Workshop on Wireless Sensor Networks (EWSN)*, Zurich, 2006.

[124] V. C. Gungor, C. Sastry, Z. Song, and R. Integlia, "Resource-aware and link quality based routing metric for wireless sensor and actor networks," in *IEEE International Conference on Communications (ICC)*, 2007, accepted, Data Available http://cc.usu.edu/ zhensong/ExperimentMeasurements.

[125] T. McEwan, "Radiolocation system having writing pen application," US Patent 6,747,599.

[126] S. Azevedo and T. Mcewan, "Micropower impulse radar" [Online]. Available: http://www.llnl.gov/str/pdfs/01_96.2.pdf.

[127] Actel Inc., "Laser range finder using Actel's axcelerator FPGA" [Online]. Available: http://www.actel.com/documents/AX_Laser_AN.pdf.

[128] T. C. Karalar, *Implementation of a Localization System for Sensor Networks*. Ph.D. dissertation, University Of California, Berkeley, 2006.

[129] R. J. Fontana and S. J. Gunderson, "Ultra-wideband precision asset location system," in *IEEE Reprinted from 2002 IEEE Conference on Ultra Wideband Systems and Technologies*, Baltimore, MD, May 2002.

[130] T. McEwan, "Short range radio locator system," US Patent 5,589,838, Aug. 1995.

[131] Pat Kinney et al., "Tg4a opening/closing report for atlanta" [Online]. Available: http://www.ieee802.org/15/pub/05/.

[132] K. Pahlavan, X. Li, and J.-P. Makela, "Indoor geolocation science and technology," *IEEE Communications Magazine*, pp. 112–118, Feb. 2002.

[133] R. A. Fleming and C. E. Kushner, "Spread spectrum localizers," US Patent 6,795,491, Dec. 11, 2000, Aether wire and location, Inc.

[134] J. Lampe and Z. Ianelli, *Introduction to Chirp Spread Spectrum (CSS) Technology*, Nanotron Technologies, Nov. 11 2003.

[135] I. Oppermann, L. Stoica, A. Rabbachin, Z. Shelby, and J. Haapola, "UWB wireless sensor networks: UWEN - a practical example," *IEEE Communications Magazine*, vol. 42, no. 12, pp. S27–S32, Dec. 2004.

[136] R. J. Fontana, "Recent system applications of short-pulse ultra-wideband (uwb) technology," *IEEE Transactions on Microwave Theory And Techniques*, vol. 52, pp. 2087–2104, 2004.

[137] T. K. Moon and W. C. Stirling, *Mathematical methods and Algorithms for signal processing*, 2nd ed. Prentice hall, 1999.

Appendices

Appendix A
Notations and Abbreviations

A.1 Common Acronyms and Notations

- LS: least square.
- WLS: weighted least square.
- FIM: Fisher information matrix.
- CRLB: Cramér-Rao lower bound.
- WSN: wireless sensor network.
- TDOA: time difference of arrival.
- TOA: time of arrival.
- TOF: time of flight.
- POA: phase of arrival.
- Vectors and matrices: vectors are typed in bold math font, such as vector $\mathbf{v}$. Matrices are indicated by capital math font, e.g., M. Note that random variable is denoted by bold capital font, e.g., $\mathbf{X}$. Thus, $\mathbf{p} \neq p \neq P \neq \mathbf{p}$, since $\mathbf{p}$ is a vector, p is a scalar, and P is a matrix. By default, all vectors are column vectors. For example, $\mathbf{p} = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} \mathbf{p}_{(1)} \\ \mathbf{p}_{(2)} \\ \vdots \\ \mathbf{p}_{(n)} \end{pmatrix}$. Note that $\mathbf{p}_{(1)}$ is the first entry of the vector $\mathbf{p}$, and $\mathbf{p}_{(1)}$ is a scalar. $\mathbf{p}_1$ is a vector. By default, M and M_1 are two difference matrices, and $M \neq M_1$.

- Subscripts: the subscripts in brackets indicate scalar entries of a matrix or vector. The subscripts without brackets are instances of variables. Capital subscripts are parts of the variable's name. So, t and t_A are different scalars. Lower case subscripts, e.g., i, j, k, are used as indices for positive integers. For example, $\mathbf{p}_{(i)}$ is the ith entry of the vector $\mathbf{p}$, and $\mathbf{p}_{(i)}$ is a scalar. Similarly, $M_{(i,j)}$ is the ith row and jth column scalar entry of matrix M. $M_{(i,:)}$ is the ith row vector of matrix M, and $M_{(:,i)}$ is the ith column vector of matrix M. $\mathbf{p}_i$ is the ith $\mathbf{p}$ vector, where $i = 1, 2, 3, \cdots$. Note that p_i is the ith p, and p_i is a scalar. Thus, $\mathbf{p}_i \neq p_i$. However, $\mathbf{p}_{(i)} = p_i$ by default, and both of them are scalars.

A.2 Notations in Chapter 2

- m: the weight of one robot.
- I: the inertia of the robot along the z axis. Note that I is a scalar. In the literature, $\mathcal{M}$ can be a 3 by 3 inertia matrix of the robot. The I in (2.1) is the entry at the 3rd row and 3rd column of $\mathcal{M}$, i.e. $\mathcal{M}_{(3,3)}$.
- l: the length of the robot's axis.
- r: wheel radius. The left and right wheels have the same radius.
- α: the yaw angle as shown in Fig. 2.1.
- (x, y): the coordinate of the center of the axis. Note that $\mathbf{x}$ is not x.
- τ_l,τ_r: the torque applied on the left and right wheel, respectively. $\tau = [\tau_l \tau_r]^T$.
- A, B, $\mathbf{x}$, τ: the parameters, states, and the control signal for a single robot.
- A_T, B_T, $\mathbf{x}_T$, τ_T: the parameters, states, and the control signal for three robots.
- b: the edge length of the robot's square chassis. It is assumed that the wheels and the axis are mounted on a square chassis.

- $\chi(t)$: Mayer state.
- $\chi_{dl}(t)$: stacks all the entries on the diagonal and below the diagonal of χ to a vector.
- Ω: the domain for valid input variables.
- $\partial\Omega$: the boundary of Ω.
- n: number of robots.
- $\mathbf{c}$: the vector of unknown parameters. $\mathbf{c} = [c_1, c_2, c_3]^T$.

A.3 Notations in Chapter 3

- n : the total number of sensors. Note that $n \neq \mathbf{n}$.
- n_i, or $\mathbf{n}_{(i)}$: the number of samples that sensor i collects in each t_S, the sampling period. Note that $\mathbf{n} = [n_1, n_2, \cdots]^T \neq n$.
- P_r: probability function.
- m: the number of parameter for identification.
- t_S: the total sampling time. In t_S, sensor i collects n_i samples.
- n_S: the max total number of samples of the whole WSN in t_S time slot.
- c_1, c_2, etc: the coefficients in sensor model.
- σ_i, $\bar{\sigma}_i$, $\tilde{\sigma}_i$: σ_i is the standard deviation of the noise of the ith sensor. $\bar{\sigma}_i$ is the standard deviation for the ith averaged sensor measurement. $\tilde{\sigma}_i$ is similar to $\bar{\sigma}_i$ but averaged over nominal sampling rates.
- $f[k]$: the value of function f for the k-th iteration.
- y_i or $\mathbf{y}_{(i)}$: the nominal value of sensor i. It is computed by the model of events.
- v_i: the noise of sensor i.

- s_i or $\mathbf{s}_{(i)}$: real value of the reading of sensor i. $s_i[k] := y_i[k] + v_i[k]$.
- $\bar{s}_i$: averaged sensor i's reading.
- $\mathcal{N}(\mu, \sigma)$: Gaussian (normal) distribution with the expectation of μ and variance of σ.
- $\mathbf{p}$: normalized sampling rate of sensor i. $\hat{\mathbf{p}}[k]$ is the optimized normalized sampling rate for the k-th iteration.
- $\mathbf{r}_i$: position of sensor i. $\mathbf{r}_i$ is assumed precisely known. In addition $\mathbf{r}_i \neq \mathbf{r}_j$, for any $i \neq j$. By default, $\mathbf{r}_i$ is a 2D vector like $\begin{pmatrix} x_i \\ y_i \end{pmatrix}$.
- $\mathbf{q}$, $\mathbf{q}^*$: $\mathbf{q}$ is the nominal position of the target. Specifically, $\mathbf{q}^*$ is the true position of the target.
- $\mathbf{1}$: an all-one vector, i.e., $[1, 1, \cdots, 1]^T$.
- ∇: gradient. For example, $\nabla_{\mathbf{q}}$ is the gradient with respect to $\mathbf{q}$.
- $\mathbf{a} \geq b$: each entry of vector $\mathbf{a}$ is no less than the scalar b. $a_i \geq b$. E.g., $\mathbf{p} \geq 0$.
- $A \succeq B$: matrix $A - B$ is positive semi-definite.

A.4 Notations in Chapter 4

- Ω: the domain for deploy mobile node and beacons.
- $\mathbf{q}_i$: the position of the ith beacon.
- $\mathbf{p}_i$: the position of the ith mobile node.
- m: number of mobile nodes.
- n: number of beacons.

Appendix B
Fisher Information for Error Bound Estimation

Fisher information is a metric to indicate the sensitivity of the measurements with respect to system parameters. The definition of Fisher information and Fisher information matrix is presented by Definition 2.3.1 in Section 2.

Following the convention in Definition 2.3.1, let $P_r(X)$ be the probability density function of the random variable X. Easy to see, if $P_r(X;\theta)$ is independent Gaussian, the sensitivity for each sensor can be computed separately.

$$\begin{aligned} l(\theta;X) &= \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{(X(\theta)-\mu)^2}{2\sigma^2}) \\ M_{(i,j)} &= E\{\frac{\partial}{\partial\theta_i}\log l(\theta;X)\frac{\partial}{\partial\theta_j}\log l(\theta;X)\} \\ M_{(i,j)} &= E\{\frac{(X-\mu)^2}{\sigma^4}\frac{\partial X}{\partial\theta_i}\frac{\partial X}{\partial\theta_j}\} \\ M_{(i,j)} &= \frac{1}{\sigma^2}\frac{\partial X}{\partial\theta_i}\frac{\partial X}{\partial\theta_j} \end{aligned}$$

Thus, (2.6), (3.5) and (4.6) are the Fisher information matrices.

Theorem B.0.1 (Cramér-Rao inequality [137] p.551) *If $t(\theta)$ is any unbiased estimator of θ based on a differentiable likelihood function, then*

$$E\{(t(\theta)-\theta)(t(\theta)-\theta)^T\} \geq M^{-1}(\theta), \tag{B.1}$$

where $M(\theta)$ is the Fisher information matrix (FIM).

Remark According to this theorem, the inverse of the FIM is the indicator for the minimum possible covariance of any unbiased estimators. In fact, if the maximum likelihood estimator, θ_{ML}, exists, the following equation holds [109]:

$$E\{(\theta_{ML}-\theta)(\theta_{ML}-\theta)^T\} = M^{-1}(\theta).$$

In other words, reducing the right hand side of (B.1) actually improves the estimator θ_{ML}, if it exists.

Appendix C
Implementations

C.1 Simulation on Trajectory Optimization

This section presents the implementation on simulations in Chapter 2. Table C.1 is an example of how to set up common parameters of the simulation and how to invoke the `riots` function. Tables C.2, C.3 and C.4 are the examples of two key functions for the `riots` function. `sys_init.m` is the initialization function for the optimization. The robots' dynamics model is formulated in `sys_h.m`, while the optimization criteria is presented in `sys_g.m`. The variable `Sensitivity` the sensitivities on some points, and it is computed by the Matlab PDE toolbox. The function `Interpolate` is a function to interpolate sensitivity values.

Table C.1: An Exemplary Setting for RIOTS Simulations

```
1  m=0.05; % weight of each robot: 0.05 kg
2  b=0.08; % edge length of each robot: 8 cm
3  l=0.08; % length between the two wheels of a robot: 8 cm
4  I=m*b*b*2/12; % inertial of a robot with respect to the z-axis
5  r=0.04; % radius of the wheel: 4 cm
6  MinU = -100; % bound of the input torque is positive and negative 100 kg-m
7  MaxU =  100;
8  s0 = [0.2; 0.1; 15/180*pi; zeros(3,1);... % The initial states of the 1st
       robot: initial position is (0.2, 0.1). Initial orientation is 15 degree and the initial
       linear and rotatory speeds are all 0.
9          0.2; 0.5; 15/180*pi; zeros(3,1); ... % The initial states of the 2nd
               robot
10         0.2; 0.8; 15/180*pi; zeros(3,1)]; % The initial states of the 3rd
               robot
11 BoundXY=1; %
12 BoundVelocity=0.5; % 0.5 m/s
13 BoundAlpha=2; % 2 rad/s
14 NumberOfParameters=3; % to observe three parameters: C1, C2, C3
15 NumberOfMayerVariable = NumberOfParameters*(NumberOfParameters+1)
       /2; % compute to number of Mayer states as the augmentation of robot's state
       variables.
16 x0 = [s0; zeros(NumberOfMayerVariable, 1)];
17 NumberOfSensorStates=6*3; % there are three robots with 6 internal states for each
       one.
18 fixed = [ones(NumberOfSensorStates, 1); zeros(
       NumberOfMayerVariable, 1)]; % Inform riots which variables are fixed thus
       should not be optimized by assigning 1 to the associated values in the "fixed" vector.
19 X0 = [x0, fixed, x0_lower, x0_upper];
20 [u_optml, x_optml, f] = riots(X0, u0, TGRID, -u_max*ones(n_ctrls
       ,1), u_max*ones(n_ctrls,1),[], [80, 0, 1], 4);
```

Table C.2: An Example of the `sys_init.m` File for RIOTS

```
function neq = sys_init(params)
global Sensitivity NumberSensors
StatesPerSensor=6; % for differentially-driven robots
if isempty(params)
    NumberControls = 2 * NumberSensors;
    NumberParameters = size(Sensitivity, 4);
    NumberStates = StatesPerSensor * NumberSensors +
       NumberParameters * (NumberParameters + 1) / 2;
    %    6 states for each differential drive robot
    %    n*(n+1)/2 is the number of unique entries in the FIM
    neq = [1 NumberStates; 2 NumberControls ];
    % neq = 1  : number of state variables.
    % neq = 2  : number of inputs
else
    global SysParams
    SysParams = params;
end
return
```

Table C.3: An Example of the `sys_h.m` File for RIOTS

```
function xdot = sys_h(neq, t, x, u)
global SysParams IndexMayerVariable NumberSensors
global b m l I r
% xdot must be a column vector with n rows.
StatesPerSensor=6; % for differentially-driven robots
NumberSensorDynamics = NumberSensors * StatesPerSensor;
NumberParameters=round(sqrt(IndexMayerVariable(end)));
x1 = x(1: StatesPerSensor: NumberSensorDynamics - 1); % x1 is the
    x-axis
x2 = x(2: StatesPerSensor: NumberSensorDynamics); % x2 is the y-axis
% the states of one robot is x=[x1 x2 alpha dx1 dx2 dalpha]'
g = InterpolateSensitivities(x1, x2, t, neq(4));
a = zeros(NumberParameters, NumberParameters);
for loop = 1: NumberSensors
    a = a + g(loop, :)' * g(loop, :);
end
A33=[-2*b/m 0 0 ; 0 -2*b/m 0; 0 0 -b*l*l/(2*I)];
A1=[zeros(3) eye(3); zeros(3) A33]; % the A matrix for one robot
A=[A1  zeros(size(A1,1), size(A1,2)*(NumberSensors-1) )];
for cnt=1:NumberSensors-1
    A=[A; zeros(size(A1,1), size(A1,2)*cnt)  A1  zeros(size(A1,1)
        , size(A1,2)*(NumberSensors-1-cnt)) ];
end
% A is comparable to A1, but for n sensors
NumberPhysicalState=NumberSensors*StatesPerSensor;
B=[];
Zero=zeros(StatesPerSensor, 2);
alpArray=x(3:StatesPerSensor:NumberPhysicalState);
for cnt=1:NumberSensors
    alp=alpArray(cnt);
    B1=[zeros(3,2);...
            r*cos(alp)/m r*cos(alp)/m;...
            r*sin(alp)/m r*sin(alp)/m;...
            -r*l/(2*I)   r*l/(2*I)];
    B=[B; repmat(Zero, 1, cnt-1) B1  repmat(Zero,1, (
        NumberSensors-cnt))];
end
xdot = [A*x(1:NumberPhysicalState)+B*u; a(MayerVariable)];
return
```

Table C.4: An Example of the `sys_g.m` File for RIOTS

```
function J = sys_g(neq, t, x0, xf)
global SysParams IndexMayerVariable

StatesPerSensor=6;
NumberSensors = round(neq(2) / 2);
NumberPhysicalState=NumberSensors*StatesPerSensor;
NumberParameters=round(sqrt(IndexMayerVariable(end)));
NumberSensorDynamics = neq(1);
FNum = neq(5);
if FNum == 1
    fim = zeros(NumberParameters, NumberParameters);
    fim(IndexMayerVariable) = xf(NumberPhysicalState+1:end); % comput FIM based on the Mayer states
    fim = fim';
    fim(IndexMayerVariable) = xf(NumberPhysicalState+1:end);
    J = -log(det(fim));
else
    error('Reference to a non-existing constraint on initial/final state')
end
return
```

C.2 Sensor Selection Demo System

C.2.1 Exemplary Matlab Code

The exemplary sensor selection code is shown in Table C.5, with the following comments on the variables.

- `NSample`: the total number of samples.
- `SenPos`: positions of all the sensors.
- `C`,`LampHeight`: parameters of the lamp.
- `sigma`: standard deviation.
- `eta`: tuning knob of the sampling rate optimization.

C.2.2 Experimental Data

The following data are collected during the experiments on our sensor selection testbed.

- The variable `est` is the estimate on the position of the lamp. The indices of `est` are associated with the testing points with the same indices. The indices of the testing points are shown in Fig. 3.23. The first row of `est`{i,j} are the manually measured positions in inches. From left to right, the three entries in the row are the x, y positions and the estimated positioning error, respectively. Since the estimated positioning errors for the manual measurement are invalid, it is always 0 for each `est`{i,j} matrix.
- The second row of each `est`{i,j} matrices is the a priori estimates on the lamp's position, with the three entries still x, y and positioning error respectively, from left to right. The values are still measured by inches.
- From the forth to the last rows of the `est`{i,j} matrices are the a posterior estimates whose formate follows the same convention of the a priori estimate in the second row.

Table C.5: Exemplary Sensor Selection Code

```
1  SenVal=SensorDataWithNoise(SelSenPos, LampPos); % collect sensor data
2  Pos0=mean(SenPos,2);
3  LSDat=struct('SenPos',SenPos,'SenVal',SenVal,'C',C,'LampHeight',
       LampHeight);
4  Optset=optimset('fmincon');
5  Pos1=fmincon(@EnergyErr, Pos0, [],[],[],[], ...
6       [0;0], [GridNumX; GridNumY]*GridSizeIn*in2cm, [], Optset,
            LSDat);
7  SenST = SensitivityComp(SenPos, Pos1, 'energy');
8  ErrM1=zeros(2,2);
9  for cnt=1:SensorNum;
10      ErrM1=ErrM1+sigma^(-2)*SenST(:,cnt)*SenST(:,cnt)' ... *round(
            NSample/SensorNum);
11      end
12 plotEllipse(Pos1,inv(ErrM1),':k');
13 p=SampOpt(Pos1, SenST, sigma, eta);        % Sampling rate optimization
14 SenInd = find(p>=ThresholdSelection);
15 SelSenP = p(SenInd);
16 SelSenPos=SenPos(:,SenInd);
17 SelSenVal= SensorDataWithNoise(SelSenPos, LampPos); % collect another
       set of sensor data
18 LSDat2=struct('SenPos',SelSenPos,'SenVal',SelSenVal,'C',C,'
       LampHeight',LampHeight);
19 Pos2=fmincon(@EnergyErr, LampPos, [],[],[],[], ...
20      [0;0], [GridNumX; GridNumY]*GridSizeIn*in2cm, [], Optset,
            LSDat2);
21 SenST2 = SensitivityComp(SelSenPos, Pos2, 'energy');
22 ErrM2=zeros(2,2);
23 for cnt=1:size(SenST2,2);
24      ErrM2=ErrM2+sigma^(-2)*SenST2(:,cnt)*SenST2(:,cnt)' * round(
            NSample*SelSenP(cnt));
25 end
26 h2ball=plotEllipse(Pos2,inv(ErrM2),'g');
27 h2pos=plot(Pos2(1),Pos2(2),'xg');
28 h2SelSen=plot(SelSenPos(1,:),SelSenPos(2,:),'rs');
29 return
```

Table C.6: Exemplary Code for Sensitivity Computation

```
1  function SenST=SensitivityComp(Pos, LampPos)
2  % Pos: the position of each sensor node
3  % LampPos: the position on the lamp
4  global C
5  SenN=size(Pos,2);
6  SenVal = SensorModel(Pos, LampPos); % nominal value, without error
7  Len=size(Pos,2);
8  SenST=zeros(2,Len);
9  for cnt=1:Len
10      SenST(:,cnt) = 2*(Pos(:,cnt)-LampPos)*(SenVal(cnt).^2)'/C;
11 end
12 return
```

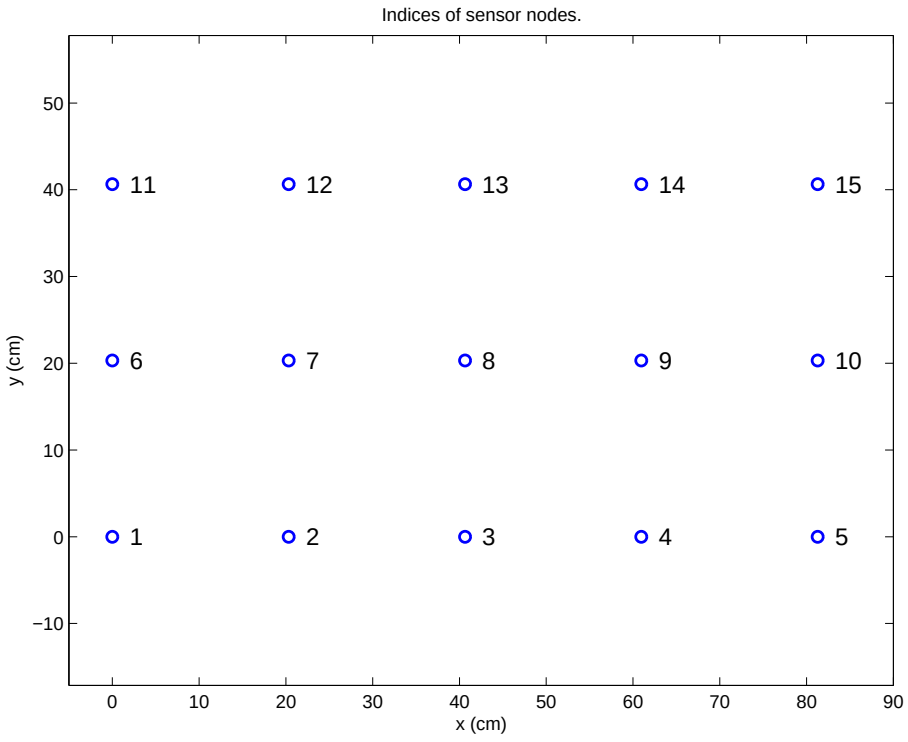


Fig. C.1: The indices for sensor nodes of the sensor selection testbed.

- The matrix `SamplingRate` are the sampling rates after the sensor selection. The rows of the matrix `SamplingRate`, from the first to the last one, are the outputs for position (1,1), (1,2), $\cdots$, (1,7), (2,1), $\cdots$, (3,7), respectively. The indices are commented on the right of each row. The columns of the matrix `SamplingRate` are the sampling rates for each sensor nodes, ordered by the indices of the sensor nodes. The indices are shown in Fig. C.1. The total sample number is 255. Due to the rounding errors, the summation of each row may not be 255 exactly.

- The matrix `LampData` is the experimental data used to plot Figs. 3.2, 3.6 and 3.34. The data characterize halogen lamps as well as the Tmote Sky sensor nodes. The columns of the matrix are the data sets collected at distances from 0 cm to 50.8 cm, with 50.8 cm apart. The rows of the matrix are ordered by the 100 trials.

```
est=cell{3,7};
est{1,1}=[
    3.8, 3, 0
    4.0474831836304,3.98640937729381,0.530539616068237
    3.69030219368631,4.53084046791177,0.245608597232329
    3.79360998078517,4.44047335771038,0.247284311837911
    3.79621500159606,4.43812979567396,0.247333701041882
    3.79900725974776,4.43551236073322,0.247378113526514
    3.80174302218333,4.43300988735921,0.247426551668037
    3.80420788178024,4.43078235024581,0.247473424908412
    3.80711507842732,4.42857523620138,0.2475153769021
    3.8097045499778,4.42617335224803,0.247561259612977
    3.81176453716081,4.42382284651583,0.24760574512124
    3.81405626047611,4.42173848606945,0.247644785685762
];

est{1,2}=[
    7.6, 3.3, 0
    8.04844648870563,4.10461796931969,0.474034178884049
    7.78988705044351,3.26877842057838,0.218798813393424
    7.85778583557264,3.70231012279228,0.221532654441194
    7.86192679005886,3.72254326525508,0.221677350012992
    7.86268122779319,3.72599101036754,0.221704707275966
    7.86299442102457,3.72896057293481,0.221729475943818
    7.86372897760968,3.7322146762133,0.221751784546935
    7.86424004209608,3.73548455575158,0.221775182307836
    7.86498119492832,3.73878126829728,0.221799368202466
    7.86546919166884,3.741852517379,0.221823167916907
    7.86618793195867,3.74495644064239,0.221845942306571
];

est{1,3}=[
    12, 2.8, 0
    12.5301623430214,3.4021758557392,0.528036383294451
    12.5294420397135,3.40524153336293,0.2424593892921 95
    12.7900441996901,3.0685666537424,0.244938500366456
    12.7928760010253,3.06620515984053,0.24495991350521
    12.8010155606284,3.04736388574673,0.245080711854237
    12.8038350664367,3.04584990716162,0.245104828795585
    12.8061397485558,3.04342702829233,0.245116957274766
    12.8078170282549,3.04069913993192,0.245134405943397
    12.8107497109872,3.03951915959153,0.245153243198329
    12.8129271275762,3.03667948673614,0.24516325582619
    12.8152575286711,3.0351250374991,0.245183184291676
];

est{1,4}=[
    15.3, 3.1, 0
    15.8323772861596,3.65729018727796,0.476469782144612
    15.7879022162631,4.68445564682164,0.226131498088571
    15.6094905438954,4.34808463598548,0.226633922743073
    15.5944614710543,4.32868423969731,0.226655668424468
    15.5922960059422,4.32588507176634,0.226658829730532
    15.5899486989545,4.32304366225173,0.226662274784185
    15.587849219258,4.32040412899407,0.22666504665521
    15.5853176349295,4.31757584665639,0.226668107160708
    15.5830726106713,4.31496905726211,0.226669969841257
    15.581110799512,4.31218027172983,0.226672212266429
    15.5791772636353,4.30947082400796,0.226676968931825
];

est{1,5}=[
    19.2, 3.2, 0
    20.2149398942565,3.92034236110996,0.497836240088391
    19.8041844512562,4.44360025388118,0.247456076130034
    19.8137229503216,4.42591174170551,0.247599608866044
    19.8352920332063,4.40095468039268,0.24800671934566
    19.8380717408332,4.3983204475248,0.248054573553278
    19.84124132214,4.39623280729016,0.248102097784305
    19.8441247815937,4.3939165385831,0.248148654855953
    19.8464684242864,4.39164588066321,0.248194386207589
    19.8475911944007,4.38871336871884,0.248234695550386
    19.8496789947044,4.38630470486163,0.248269122587239
    19.8514765997059,4.38371663964075,0.248308105433175
];

est{1,6}=[
    23,3.3, 0
    23.4269705135168,4.12908722018588,0.474318308701361
    24.1851374508688,3.47794833112922,0.262776502012932
    24.2056483211575,3.9096396046454,0.262622132598746
    24.2264813432486,3.94906325937053,0.263038148207259
    24.2366940949596,3.96696274390946,0.263267420558916
    24.2384594543067,3.96973520112174,0.263312607391327
    24.2399539702703,3.97230541432262,0.263353172150807
    24.2401667566297,3.97419000623648,0.263387378212243
    24.2397707185921,3.9753254663463,0.26339083429375
    24.2389034355644,3.97659349865715,0.26338005871795
    24.2382601756784,3.97792301541945,0.263357603249006
];

est{1,7}=[
    26.8, 3.6, 0
    27.5725386207448,4.80468523153953,0.526022594967061
    27.5733024278611,4.66731049947225,0.245229750431927
    27.5738603104731,4.66361680980554,0.24517038515931
    27.4633925778482,5.07147056596402,0.25262593132801
    27.4618030342764,5.07487018164453,0.252701019037374
    27.4601343301208,5.07826269715765,0.25277433985669
    27.4583153821469,5.08130138199575,0.252847446347764
    27.4568222667089,5.0840248556857,0.252912088257293
    27.4553609680044,5.08668047409301,0.252970565270152
    27.4778290153921,5.14803823223532,0.258296660822031
    27.4759928106581,5.15566491531472,0.25853433991981
];

est{2,1}=[
    3.8, 7.6, 0
    5.19247472077239,8.38626785551865,0.49245825836639
    5.43855852814086,6.49575163166598,0.238726884130604
    5.5300084799918,7.10290308856285,0.243287696986985
    5.52910155290043,7.10647917238645,0.243327004516333
    5.52867208536207,7.1103902502845,0.243364510889071
    5.52815725289167,7.11405857387911,0.243402383757097
    5.52716529095717,7.11732101151381,0.243438544988974
    5.52659728528346,7.12070011516655,0.243473726119788
    5.52561928174833,7.12390084768421,0.243507584298037
    5.52431571702406,7.12715308779106,0.243542151471669
    5.52338333241307,7.1303280332901,0.243579043105776
];

est{2,2}=[
    7, 6.9, 0
    7.10351390415587,8.32211001953104,0.464904533379217
    7.09131262087992,7.2554386028169,0.23626064828166
    7.04588829045246,7.71837183698774,0.239454026071859
    7.04518147964838,7.72218286752515,0.239476836274018
    7.04446875842448,7.72594293632048,0.239497629139189
    7.04337383401 72,7.7419728070872,0.239575090596243
    7.04253675693438,7.74545167873637,0.239593382648254
    7.04124341296261,7.7487642276587,0.239611972979637
    7.04031124217408,7.75228932892047,0.239631773136892
    7.03906564388669,7.75552739139764,0.239650603090459
    7.03768523759128,7.75860358114261,0.23966950681079
];

est{2,3}=[
    11.2, 7.3, 0
    12.7244546498609,7.91581685936649,0.474274559157398
    12.5944366152738,7.85704092796102,0.231102058331098
    12.591537434999,7.85563722171299,0.23106004344745
    12.5886320314792,7.85404592076225,0.231025323588074
    12.585908215043,7.85271958407035,0.23098980925528
    12.5827664259719,7.85118631665943,0.230957258695373
    12.5797194670246,7.84969279228447,0.230919757107458
    12.5772644430512,7.84847946835278,0.230883419758822
    12.5749837136277,7.84706766596286,0.23085414324467
    12.5726716650092,7.8459220084 59,0.230825809168539
    12.5708351814446,7.84482436471876,0.230798297213902
];

est{2,4}=[
    15.5, 7.8, 0
    14.4356781955664,9.136755626350 47,0.470559229881083
    14.1454778673083,9.01483418485509,0.248127537728326
    14.143358909858,9.01154854031136,0.248074608122404
    14.368104182344,8.95601449695197,0.249753138574824
    14.3718628056138,8.95582478389369,0.249794249846502
    14.3749876242079,8.95559282408547,0.24983990686404
    14.3780523281553,8.95513520343895,0.249877065358886
    14.3808979800278,8.95500626963005,0.249910451350276
    14.3840088741955,8.95469284110166,0.249945833614049
    14.386980268267,8.9544311250481,0.249982260932188
    14.3898950030945,8.95454994472526,0.250017790346692
];

est{2,5}=[
    19.5, 7.5, 0
    20.8715408008781,7.88007813823719,0.480274061779725
    19.7766467180037,7.90031835501856,0.230224568977455
    19.7740648486673,7.90056936338291,0.230205511946842
    20.2711164479792,7.59689858755187,0.23118118443546
    20.2740859061849,7.59465458271047,0.231188390585385
    20.2764833105639,7.59219954845736,0.231197205167541
    20.2780255479161,7.58970730319232,0.231200907008055
    20.2795269056123,7.58726615144305,0.231198415148742
    20.2814949114178,7.5846904969913,0.231195982858905
    20.2836921022796,7.58243955755335,0.231196234462091
    20.2859220869828,7.58044548217516,0.231199936578724
];

est{2,6}=[
    23.5, 7, 0
    23.2291287259428,8.52600994778599,0.464769169249906
    24.1919734328888,6.19244286700353,0.234933041834163
    24.2185858725472,6.49483621453919,0.234684813943579
    24.2202262258969,6.49799063581697,0.234688540982829
    24.2218764542288,6.50078961639374,0.234694268612303
    24.2232078501157,6.50397743826536,0.234700348734883
    24.2247957811767,6.50720445733702,0.234704580980775
    24.2261369592064,6.51034059285691,0.234710095874418
    24.2274975456499,6.51289674726909,0.234714484578506
    24.2295032681051,6.51455788136657,0.234719417957237
    24.2314938776745,6.51638141608923,0.234728222249876
];

est{2,7}=[
    27.3, 8, 0
    28.5928281003494,8.51380349863577,0.46953183590475
    28.7191276728031,7.26318106148668,0.219098408199878
    28.2782765639044,7.64465664621348,0.220281684340674
    28.2761619143566,7.64781956109518,0.220293441637172
    28.2745972443569,7.65069776320583,0.220304983306441
    28.2726018200144,7.65376146955409,0.220314151541114
    28.2706706587754,7.65677468057445,0.220325323920195
    28.2688017692495,7.65973831911525,0.220336289276854
    28.2669932189114,7.66265328743853,0.220347050895101
    28.2648637221324,7.66555046382949,0.220357612057879
    28.2632847975459,7.66816781891551,0.22036953845711
];

est{3,1}=[
    3.5, 11.7, 0
    3.09687762311607,12.5689839394288,0.555780502598974
    3.92359954309961,12.1001918512246,0.238470752046229
    3.94391559619304,12.0982973141513,0.238324441753265
    3.98799444479162,12.1067602577344,0.238054228823887
    4.02573235673257,12.120021143127,0.237853616152455
    4.06264798146971,12.1379265487523,0.237685400797833
    4.10684749231686,12.1647741912006,0.237518396400454
    4.1360872669641,12.1845313243231,0.237425067643203
    4.20268688562826,12.2259465660745,0.237231572809417
    4.417049431429,12.3516375693853,0.236817116445147
    4.41595406306127,12.3503609142533,0.236811748000578
];

est{3,2}=[
    8, 11.2, 0
    8.48182696704214,12.3442172791527,0.494885545009503
```

```
    8.39393545173675,12.3407335754648,0.2639348497126 07
    8.39008436610658,12.3407959058925,0.263912542174231
    8.37423391186283,12.340764581803,0.263822614654134
    8.37042123269111,12.3411751518556,0.263799878451527
    8.36677627616588,12.3413320744812,0.263772673445137
    8.3631984259023,12.3426464193286,0.263749387885431
    8.35969718737714,12.3435065379197,0.263712345331416
    8.35627277542459,12.3439369450584,0.263681158070238
    8.35262690955769,12.3438715710763,0.26365551464843
    8.34928357483886,12.3443114512877,0.263634400596631
];

est{3,3}=[
    12.1, 10.8, 0
    12.1545258924819,12.14087220982,0.51969478800259
    12.4587856607152,11.9086271237921,0.241023229865354
    13.086612409275,12.099189461058,0.250191524091398
    13.0164085332725,12.3474567130655,0.248734125918859
    13.0628151138377,12.3248090794699,0.24946032031902
    12.9751933397787,12.5444239460777,0.247800249333395
    13.2274154863334,12.3665948980713,0.252343481984132
    13.188742043957,12.5074728986228,0.251341392946569
    13.1565165908766,12.5726132698614,0.250615013478188
    13.155106399523,12.5746932480242,0.250576550664049
    13.1561131076997,12.5748755752769,0.250546945157326
];

est{3,4}=[
    15.2, 11.6, 0
    14.5526142604682,12.2590462449459,0.482024832778939
    13.9351282158724,12.2989834503444,0.284945766294924
    14.142758115211,12.2412202647842,0.298551486143719
    14.146009875663,12.2403092631596,0.298838121625628
    14.149216411208,12.2394253841412,0.299083219504926
    14.1520822111325,12.2392461100974,0.299325613820961
    14.1549064064729,12.2390644044544,0.299540326352619
    14.1578374380442,12.2385410682676,0.299752537720442
    14.1607271037254,12.2380296811226,0.2999748162573
    14.1636071971757,12.2379378730859,0.30019456715541
    14.1665927789055,12.237500079389,0.300412446316843
];

est{3,5}=[
    19.6, 11.2, 0
    20.1759974569403,11.8348526478546,0.525240604622037
    20.2625320853757,11.8436422478968,0.239340098732339
    20.2663230715709,11.8435260899923,0.2393690472123
    20.270001607495,11.8433639373404,0.239397353851925
    20.2738216285464,11.8433836763175,0.239425031959896
    20.0582650101058,12.0042005207837,0.252822911302391
    20.0593964397805,12.0028386822792,0.252843778237171
    20.0605120159667,12.0015030904996,0.252860489901919
    20.0616118527779,12.0001931721269,0.252876896153131
    20.0629365807236,11.9991452437586,0.25289300271723
    20.063996914362 3,11.9978689664111,0.252908925627841
];

est{3,6}=[
    23.8, 11.5, 0
    25.4209886143985,12.2126252022009,0.500049229713879
    25.4333491965396,12.0772573192592,0.2474726875701
    25.4602379287259,12.2715976242901,0.24656359758152
    25.4609098737801,12.275270820689,0.246547183529 42
    25.4613058889895,12.2786174745469,0.246530483425011
    25.4616768785831,12.2817953217742,0.246516357548666
    25.4619037089963,12.284459772763,0.246503020026926
    25.4620899920182,12.2877058887874,0.246492272070595
    25.4622770874241,12.2900742340604,0.246479644524678
    25.4625684007942,12.292676863969,0.246470221876805
    25.4626979111883,12.2955108497618,0.246459498137656
];

est{3,7}=[
    26.4, 11.5, 0
    27.3900942961187,11.601454487046,0.523021807052 73
    27.3883920167765,11.6024585113237,0.244532340543695
    26.8275215897316,11.4284901254502,0.254476312425801
    26.8180378780527,11.4209831453314,0.254681941268781
    26.8071181328237,11.4159366048291,0.254906896641411
    26.8041412341345,11.4149053971816,0.254984741434298
    26.8009465493263,11.4130126194084,0.255051584540131
    26.7981575167954,11.4115767621728,0.255120500091348
    26.7957080954625,11.4099972140854,0.255181611407412
    26.793318451826,11.4084582394704,0.255234152693239
    26.7909871520897,11.4069588622595,0.255285495600224
];

SamplingRate=[
 84, 84, 0, 0, 0, 0, 84, 0, 0, 0, 0, 0, 0, 0, 0 %1,1
 0, 0, 0, 0, 0, 0, 59, 117, 77, 0, 0, 0, 0, 0, 0 %1,2
 0, 74, 125, 0, 0, 0, 0, 0, 55, 0, 0, 0, 0, 0, 0 %1,3
 0, 100, 121, 33, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 %1,4
 0, 0, 84, 84, 0, 0, 0, 0, 0, 84, 0, 0, 0, 0, 0 %1,5
 0, 0, 2, 0, 0, 0, 0, 0, 125, 126, 0, 0, 0, 0, 0 %1,6
 0, 0, 0, 19, 0, 0, 0, 0, 0, 126, 109, 0, 0, 0, 0 %1,7
 0, 0, 0, 0, 0, 0, 84, 84, 0, 0, 0, 0, 84, 0, 0 %2,1
 0, 0, 0, 0, 0, 0, 84, 84, 0, 0, 0, 0, 84, 0, 0 %2,2
 0, 0, 88, 0, 0, 0, 0, 0, 121, 0, 0, 0, 0, 45, 0 %2,3
 0, 0, 0, 0, 0, 0, 0, 32, 114, 0, 0, 0, 0, 108, 0 %2,4
 0, 0, 0, 100, 0, 0, 0, 0, 0, 123, 0, 0, 0, 0, 31 %2,5
 0, 0, 0, 0, 0, 0, 0, 0, 0, 84, 84, 0, 0, 0, 0, 84 %2,6
 0, 0, 0, 0, 0, 0, 0, 0, 0, 112, 0, 0, 0, 0, 0, 67, 74 %2,7
 0, 0, 0, 0, 0, 0, 88, 0, 0, 0, 0, 0, 124, 41, 0, 0, 0 %3,1
 0, 0, 0, 0, 0, 0, 0, 84, 0, 0, 0, 0, 0, 84, 84, 0, 0 %3,2
 0, 0, 0, 0, 0, 0, 0, 0, 0, 93, 0, 0, 0, 0, 34, 127, 0, 0 %3,3
 0, 0, 0, 0, 0, 0, 0, 0, 0, 123, 0, 0, 0, 0, 0, 120, 11, 0 %3,4
 0, 0, 0, 0, 0, 0, 0, 0, 0, 44, 127, 0, 0, 0, 0, 0, 83, 0 %3,5
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 33, 0, 0, 0, 0, 0, 116, 105 %3,6
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 126, 22, 0, 0, 0, 0, 105, 0 %3,7
];

LampData=[...
6799, 5773, 4901, 3900, 3356, 2758, 2459, 2423, 2032, 2160, 2178;...
6750, 5639, 4797, 3967, 3350, 2783, 2459, 2416, 2056, 2166, 2185;...
6738, 5780, 4919, 3918, 3363, 2795, 2478, 2416, 2026, 2148, 2197;...
6787, 5749, 4852, 3887, 3381, 2777, 2441, 2429, 2020, 2142, 2215;...
6805, 5639, 4772, 3918, 3405, 2770, 2447, 2429, 2026, 2166, 2178;...
6768, 5798, 4852, 3979, 3375, 2819, 2484, 2404, 2020, 2166, 2191;...
6750, 5755, 4931, 3967, 3344, 2764, 2484, 2404, 2032, 2166, 2191;...
6787, 5651, 4797, 3900, 3332, 2795, 2465, 2404, 2038, 2142, 2172;...
6744, 5682, 4797, 3881, 3356, 2801, 2465, 2435, 2020, 2142, 2178;...
6726, 5798, 4901, 3912, 3399, 2777, 2459, 2410, 2038, 2166, 2203;...
6750, 5670, 4772, 3967, 3344, 2783, 2484, 2410, 2020, 2166, 2191;...
6781, 5651, 4840, 3955, 3363, 2789, 2459, 2416, 2020, 2142, 2191;...
6762, 5755, 4925, 3869, 3369, 2764, 2478, 2429, 2038, 2136, 2209;...
6726, 5798, 4846, 3881, 3417, 2789, 2465, 2410, 2050, 2172, 2209;...
6726, 5712, 4766, 3918, 3399, 2795, 2459, 2410, 2026, 2160, 2203;...
6750, 5682, 4888, 3973, 3375, 2770, 2453, 2423, 2020, 2172, 2209;...
6707, 5767, 4791, 3942, 3350, 2801, 2496, 2410, 2050, 2154, 2209;...
6732, 5780, 4797, 3894, 3363, 2801, 2478, 2435, 2056, 2178, 2197;...
6756, 5694, 4925, 3875, 3405, 2777, 2447, 2410, 2044, 2142, 2185;...
6774, 5633, 4888, 3948, 3399, 2783, 2453, 2416, 2026, 2154, 2203;...
6707, 5712, 4785, 3967, 3369, 2770, 2465, 2423, 2044, 2154, 2191;...
6732, 5798, 4919, 3918, 3356, 2801, 2465, 2423, 2056, 2160, 2185;...
6781, 5780, 4779, 3857, 3369, 2764, 2484, 2423, 2062, 2172, 2197;...
6774, 5676, 4815, 3875, 3411, 2770, 2471, 2423, 2044, 2148, 2178;...
6726, 5651, 4901, 3936, 3405, 2795, 2453, 2398, 2050, 2172, 2197;...
6738, 5706, 4797, 3955, 3363, 2770, 2465, 2423, 2032, 2166, 2215;...
6756, 5737, 4797, 3912, 3363, 2783, 2459, 2404, 2026, 2160, 2197;...
6719, 5639, 4882, 3851, 3405, 2770, 2465, 2435, 2026, 2166, 2203;...
6713, 5633, 4919, 3875, 3417, 2770, 2465, 2429, 2044, 2142, 2209;...
6774, 5731, 4821, 3900, 3393, 2807, 2453, 2392, 2056, 2142, 2209;...
6719, 5786, 4840, 3948, 3363, 2795, 2447, 2416, 2044, 2160, 2197;...
6707, 5609, 4754, 3887, 3405, 2801, 2496, 2410, 2026, 2148, 2215;...
6750, 5682, 4833, 3869, 3387, 2758, 2484, 2404, 2062, 2160, 2191;...
6774, 5786, 4846, 3918, 3363, 2777, 2478, 2404, 2056, 2148, 2197;...
6726, 5761, 4803, 3961, 3375, 2807, 2484, 2386, 2044, 2172, 2209;...
6701, 5645, 4919, 3948, 3411, 2770, 2484, 2423, 2056, 2166, 2197;...
6719, 5633, 4901, 3900, 3387, 2783, 2465, 2429, 2032, 2172, 2203;...
6756, 5798, 4876, 3875, 3332, 2770, 2453, 2423, 2056, 2148, 2185;...
6744, 5633, 4925, 3912, 3399, 2770, 2465, 2404, 2038, 2185, 2191;...
6689, 5700, 4809, 3961, 3405, 2770, 2459, 2416, 2038, 2154, 2215;...
6707, 5810, 4760, 3924, 3356, 2764, 2465, 2398, 2044, 2172, 2215;...
6756, 5731, 4827, 3869, 3363, 2795, 2441, 2404, 2050, 2148, 2185;...
6750, 5798, 4919, 3851, 3405, 2795, 2447, 2416, 2026, 2178, 2209;...
6707, 5737, 4888, 3900, 3430, 2795, 2478, 2404, 2050, 2166, 2209;...
6701, 5633, 4785, 3924, 3375, 2795, 2459, 2386, 2044, 2154, 2191;...
6750, 5682, 4785, 3948, 3363, 2770, 2471, 2429, 2056, 2148, 2191;...
6713, 5780, 4913, 3930, 3411, 2789, 2441, 2410, 2062, 2166, 2185;...
6695, 5633, 4901, 3881, 3399, 2801, 2478, 2404, 2056, 2178, 2215;...
6726, 5670, 4791, 3863, 3338, 2764, 2478, 2410, 2038, 2160, 2209;...
6726, 5773, 4913, 3869, 3375, 2807, 2490, 2416, 2044, 2166, 2215;...
6695, 5657, 4901, 3894, 3411, 2795, 2441, 2410, 2050, 2166, 2221;...
6707, 5633, 4797, 3930, 3399, 2758, 2484, 2410, 2038, 2154, 2185;...
6750, 5725, 4803, 3961, 3363, 2783, 2484, 2410, 2050, 2172, 2209;...
6726, 5773, 4925, 3900, 3350, 2777, 2465, 2398, 2050, 2154, 2209;...
6713, 5725, 4901, 3863, 3417, 2777, 2478, 2398, 2032, 2160, 2203;...
6756, 5639, 4797, 3887, 3405, 2770, 2453, 2435, 2038, 2160, 2215;...
6707, 5725, 4901, 3930, 3338, 2801, 2478, 2398, 2062, 2160, 2191;...
6683, 5786, 4797, 3948, 3399, 2770, 2471, 2398, 2038, 2166, 2209;...
6726, 5700, 4772, 3912, 3405, 2807, 2471, 2386, 2032, 2185, 2197;...
6744, 5590, 4901, 3845, 3356, 2801, 2465, 2429, 2050, 2172, 2191;...
6701, 5615, 4901, 3881, 3363, 2795, 2459, 2392, 2056, 2166, 2215;...
6726, 5700, 4791, 3900, 3393, 2758, 2484, 2404, 2038, 2178, 2191;...
6750, 5761, 4882, 3955, 3350, 2789, 2459, 2398, 2044, 2172, 2209;...
6689, 5731, 4895, 3948, 3387, 2770, 2459, 2410, 2044, 2166, 2209;...
6689, 5633, 4797, 3875, 3417, 2807, 2471, 2404, 2062, 2178, 2209;...
6732, 5657, 4785, 3839, 3381, 2795, 2459, 2404, 2069, 2185, 2221;...
6701, 5761, 4901, 3851, 3393, 2832, 2465, 2404, 2038, 2166, 2221;...
6689, 5786, 4760, 3936, 3424, 2801, 2478, 2410, 2056, 2178, 2197;...
6726, 5688, 4772, 3900, 3393, 2801, 2465, 2386, 2062, 2185, 2203;...
6713, 5609, 4870, 3851, 3350, 2777, 2496, 2410, 2038, 2185, 2203;...
6677, 5633, 4901, 3863, 3363, 2777, 2484, 2404, 2032, 2166, 2215;...
6701, 5712, 4766, 3894, 3411, 2819, 2465, 2423, 2056, 2160, 2227;...
6738, 5792, 4901, 3924, 3350, 2825, 2453, 2386, 2050, 2185, 2203;...
6738, 5731, 4809, 3863, 3356, 2770, 2459, 2410, 2056, 2166, 2191;...
6701, 5651, 4772, 3851, 3381, 2801, 2478, 2392, 2044, 2154, 2209;...
6671, 5621, 4913, 3900, 3430, 2795, 2453, 2410, 2056, 2178, 2215;...
6719, 5706, 4797, 3948, 3387, 2783, 2453, 2429, 2062, 2185, 2215;...
6744, 5633, 4913, 3875, 3363, 2777, 2471, 2416, 2038, 2191, 2203;...
6707, 5645, 4870, 3900, 3350, 2807, 2459, 2392, 2032, 2166, 2203;...
6671, 5657, 4779, 3948, 3411, 2770, 2441, 2429, 2056, 2160, 2221;...
6707, 5633, 4827, 3948, 3369, 2770, 2447, 2410, 2056, 2166, 2197;...
6738, 5676, 4919, 3863, 3332, 2783, 2465, 2410, 2044, 2178, 2227;...
6701, 5798, 4809, 3948, 3375, 2783, 2484, 2404, 2038, 2166, 2227;...
6665, 5627, 4791, 3942, 3399, 2764, 2478, 2392, 2038, 2172, 2209;...
6707, 5621, 4888, 3875, 3411, 2764, 2465, 2410, 2044, 2160, 2227;...
6744, 5712, 4766, 3845, 3369, 2819, 2490, 2404, 2038, 2166, 2227;...
6707, 5761, 4840, 3918, 3375, 2813, 2459, 2410, 2044, 2185, 2221;...
6677, 5609, 4919, 3967, 3405, 2795, 2459, 2410, 2069, 2191, 2221;...
6726, 5657, 4821, 3936, 3356, 2807, 2484, 2392, 2069, 2178, 2209;...
6744, 5731, 4779, 3845, 3399, 2813, 2453, 2404, 2032, 2172, 2215;...
6683, 5798, 4876, 3851, 3405, 2783, 2465, 2429, 2069, 2160, 2197;...
6683, 5755, 4913, 3918, 3405, 2777, 2471, 2404, 2038, 2185, 2233;...
6738, 5670, 4803, 3948, 3338, 2795, 2459, 2410, 2038, 2191, 2209;...
6677, 5609, 4913, 3851, 3375, 2807, 2459, 2404, 2056, 2197, 2221;...
6701, 5645, 4779, 3900, 3430, 2813, 2471, 2410, 2069, 2166, 2197;...
6732, 5725, 4840, 3967, 3381, 2795, 2465, 2416, 2056, 2178, 2233;...
6677, 5767, 4943, 3912, 3356, 2777, 2471, 2410, 2044, 2172, 2215;...
6695, 5700, 4864, 3845, 3381, 2783, 2471, 2404, 2075, 2191, 2233;...
6744, 5621, 4772, 3887, 3411, 2801, 2465, 2392, 2044, 2185, 2233;...
6726, 5627, 4858, 3942, 3399, 2801, 2478, 2410, 2050, 2185, 2227];
```