Spatiotemporal Relationship Aided Field Estimation & Route Planning for Large Scale Mobile Cyber Physical Systems

Submitted in partial fulfillment for the requirements for
the degreee of
Doctor of Philosophy
in
Electrical & Computer Engineering

Xinlei Chen

B.S., Electronic Engineering, Tsinghua University
M.S., Electronic Engineering, Tsinghua University

Carnegie Mellon University
Pittsburgh, PA

May 2018

Acknowledgments

First, I would like to thank my advisor Professor Pei Zhang. Thank you for all your support on my research, my life and my future career development. Thank you for bring me into the exciting research world. Thank you for giving me the freedom to try different research ideas and keeping me on a cohesive research path. Thank you for continuously challenging me and pushing me, which makes me improve a lot on both research and life. I feel very fortunate to have had you as my mentor and advisor.

A special thanks to all my other committee members. Thank you Professor Jelena Kovačević for your suggestions on many unclear parts in my thesis. Thank you Professor John Paul Shen for many discussions. Your insightful comments help me improve my thesis and presentation a lot. Thank you Professor Lin Zhang for your support on several research projects. Thank you Professor Hae Young Noh for every valuable feedback from every weekly group meeting.

Many thanks to all my lab-mates and friends at Carnegie Mellon University: Zheng Sun, Aveek Purohit, Frank Mokaya, Shijia Pan, Carlos Ruiz, Adeola Bannis, Amelie Marie Bonde, Susu Xu, Mostafa Mirshekari, Jonathon Fagert, Abhinav Jauhri, Jun Han, Bing Liu, Ming Zeng, Guanlin Chao, Yuan Tian, Xiao Wang, for all the help, advise and fun.

Many thanks to all collaborators and friends at Tsinghua University: Yong Li, Weiwei Jiang, Xinyu Liu, Xiangxiang Xu, Tian Zhou, Yue Zhang, Weixi Gu, Sihan Zeng, Yu Wang, for all the help, advise and fun.

Most importantly, I want to thank all my family members. Thank you for all the support and encouragement. I am very luck to have you.

Finally, this thesis work was partially supported by the U.S. National Science Foundation under awards CNS1149611 and CNS1135874, by DARPA under grant D11AP00265, by Intel Inc. and by Nokia Inc. I greatly appreciate the financial support.

Abstract

A large scale mobile Cyber Physical System (CPS), which consists of a large number of mobile devices interacting with each other and the physical environment, is an integrated system of computation, networking and physical processes. In recent years, CPSs have gradually transformed how people interact with and control the physical world in many domains: agriculture, transportation, health-care, manufacturing, energy, defense, aerospace, buildings, etc.

A large scale mobile CPS understands the physical world by sensing data to estimate the status of physical fields. This thesis focuses on two major tasks of large scale mobile CPSs: **field estimation** and **route planning**. The task of field estimation is to use sensing data of physical fields to estimate two statuses: 1) *physical field*: a physical quantity, represented by a number or tensor, that has a value for each point in space and time, such as air pollution, temperature, moisture, noise, traffic, etc; 2) *system status*: the conditions of the system's mobile devices such as location, mobility, sensing accuracy, etc. The task of route planning is to design the routes for mobile devices in the system for data collection, which guarantees field estimation to achieve application specific accuracy.

However, the real system faces two main challenges: **lacking dense coverage and lacking even distribution** of data collection. *A dense coverage* requires that the percentage of the overall space and time period being sensed by the mobile devices in the system should exceed a minimum number. *An even distribution* requires the information entropy of data distribution over space and time should exceed a minimum number. To improve the coverage and evenness of the data distribution, route planning designs routes for mobile devices to make sure that they sense data at designated locations and times. Since route planning relies on field estimation, especially system status estimation (e.g. locations of mobile devices), inaccuracy from field estimation deteriorates route planning performance. In addition, many real-world systems are semi-controllable. Only a fraction of total mobile devices follow the suggested routes from

the system. This leads to two challenging problems: how to select mobile devices for route planning and how to design routes for the selected mobile devices.

The thesis presents a spatiotemporal relationship aided framework for large scale mobile CPSs, which incorporates a new *spatiotemporal relationship analysis layer* to address the challenges of lacking dense coverage and lacking even distribution of data collection. By utilizing the spatiotemporal relationships of physical field and system status in the *spatiotemporal relationship analysis layer*, which are discussed in Section 2, models and algorithms are designed to improve the performance of major system tasks: field estimation (physical field and system status) and route planning.

I deploy real testbed experiments and extensive simulations with real world collected data to validate the system design. As a part of the evaluation for uncontrolled to controlled motion aspects of our system, air pollution sensors are deployed on the taxi-based testbed to collect data in the city of Shenzhen for 2 years in collaboration with Tsinghua University. In addition, a swarm of 8 micro aerial vehicles are deployed in an indoor environment for autonomous navigation. The results show incorporating the *spatiotemporal relationship analysis layer* can achieve 2.1× and 6× error reduction on physical field and system status estimation and 3× improvement on route planning. This illustrates the potential of the *spatiotemporal relationship analysis layer* to improve the performance of field estimation and route planning in large scale mobile CPSs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Large Scale Mobile Cyber Physical System

Cyber physical systems(CPSs) are smart systems that integrate the capabilities of computation, communication, sensing and actuation. The embedded computers in CPSs utilize these capabilities to monitor and control the physical world, while the feedback from the physical world adversely affects the computation of CPSs [2]. In recent years, CPSs have gradually transformed how people interact with and control the physical world in many critical areas, such as emergency response, city management, smart manufacturing, defense and homeland security, aerospace, energy supply and use, etc [3, 4]. The research of CPSs has aroused significant attention all over the world. According to the President's Committee of Advisors on Science and Technology (PCAST), CPSs have been called a national research priority of the United States [5]. By implementing CPSs to achieve just a one percent improvement in efficiency, large amounts of cost can be saved: $30 billion in aviation sector fuel costs, $66 billion in power generation, $63 billion in health care and $27 billion in freight rail costs over a 15-year period [6]. In addition, CPSs are also listed as one of the most important research areas in European Union Horizon 2020 Program [7]. It is estimated that German manufacturing will boost its gross value by 267 billion euros by 2025 with the integration of CPSs [8].

Large scale mobile CPSs, which consist of a large number of mobile devices interacting with the physical world, are a prominent subcategory of CPSs. Such systems utilize the high mobility and large number of mobile devices to collect

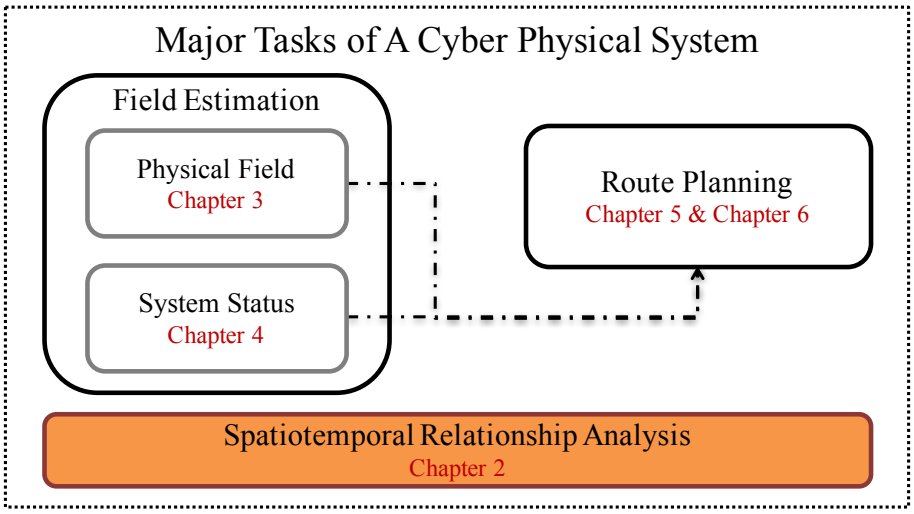


Figure 1.1: This figure shows two major tasks of CPSs: field estimation and route planning. The task of field estimation consists of physical field estimation and system status estimation. The task of route planning relies on the outputs from field estimation. All tasks are supported by the *spatiotemporal relationship analysis layer*.

information over large space and time to understand the physical world.

An effective way to understand the physical world is to estimate the status of physical fields. I focus on using sensing systems to achieve this estimation, hence it is important to estimate the status of the system to ensure the operation of each mobile device for sensing. Due to the mobility of the system, designing routes of mobile devices is a useful way to make sure that mobile devices sense at designated locations and times.

This thesis focuses on two major tasks of large scale mobile CPSs: **field estimation** and **route planning**. The task of field estimation is to use sensing data of physical field for estimation of two statuses: 1) *physical field*: a physical quantity, represented by a number or tensor, that has a value for each point in space and time, such as air pollution, temperature, moisture, noise, traffic, etc. The thesis focuses on physical fields that continuously change over time and space; 2) *system status*: the conditions of the mobile devices, which are related to the system operation, such as location, mobility, sensing accuracy, etc. The task of route planning is to design the routes for mobile devices in the system for data collection, which is necessary for field estimation to achieve application specific accuracy. Knowing the system status and physical field gives important information for designing routes. Therefore, the

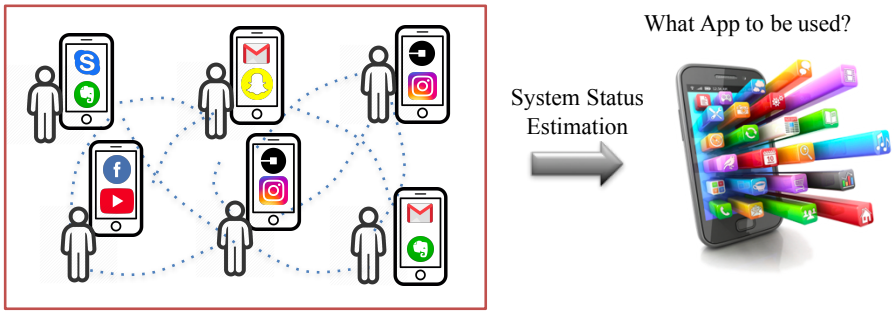


Figure 1.2: This figure shows an example of applying a mobile phone based CPS for understanding human interests and behaviors. This is an implementation of estimating system status since human beings have been regarded as an important component in the loop of CPS. The mobile phones collect data from various Apps in order to record the users' interests.

task of route planning relies on the outputs from field estimation.

Based on the system's *controllability over the routes of mobile devices*, large scale mobile CPSs can be categorized into three types: **uncontrollable**, **controllable** and **semi-controllable**. The uncontrollable system has no control over the routes of mobile devices. No mobile devices take route suggestions from the system. In contrast, the controllable system has full control over the routes of mobile devices. All mobile devices exactly follow the suggested routes from the system. The semi-controllable system has partial control over the routes of mobile devices. Only a fraction of total mobile devices follow the suggested routes from the system. It is noticed that 1) field estimation is independent of the system type, since the type only decides the system's controllability over the routes of mobile devices, thus only affecting route planning; 2) there is no route planning for the uncontrollable system, since the uncontrollable system has no control over the routes of mobile devices.

1.2 Example Applications

In order to show the combinations of the system task and the system type, this section introduces four application examples. Two examples are about system status estimation and physical field estimation when the system does not control the routes of mobile devices. Another two examples are about route planning for controllable and semi-controllable systems. The system dispatches mobile devices to sense data, which improves the accuracy of field estimation. The first and second estimate human

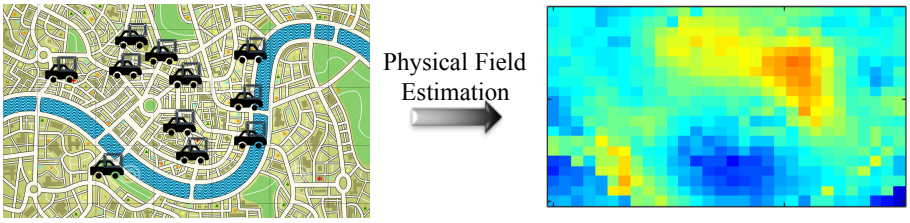


Figure 1.3: This figure shows an example of utilizing vehicle based CPS for urban air pollution sensing. This is an implementation of estimating physical field. The vehicle based CPS collects data to estimate real-time, fine-grained city-scale air pollution. It benefits both residents and city administrations to understand air pollution in their immediate environment.

behavior with mobile phone based CPS and air pollution with vehicle based CPS. The third and fourth plan routes for controllable drone based CPS and semi-controllable vehicle based CPS. The technique details of the four examples will be introduced in Chapter 3, 4, 5 and 6 respectively.

1.2.1 System Status Estimation for Large Scale Mobile CPS

In recent years, human beings have been regarded an important components in the loop of CPSs [9]. Human behaviors are regarded as an important system status to be estimated in large scale mobile CPSs. Mobile phones are effective ways to understand human behaviors since people spend over 4 hours a day on their mobile phones, which collect various types of data from users [10]. Figure 1.2 shows an example of adopting the mobile phone based CPS to estimate system status, i.e. understanding human behavior and interest such as mobile application (App) usage pattern. Understanding users' mobile App usage pattern benefits a variety of commercial applications such as precise bandwidth allocation, targeted advertisement, etc [11]. The details of estimating system status for the large scale mobile phone based CPS can be found in Chapter 3.

1.2.2 Physical Field Estimation for Large Scale Mobile CPS

Figure 1.3 gives an example of adopting vehicle based CPS to understand real-time fine-grained urban information, including air pollution, noise, traffic condition etc. Especially, air pollution has become a major public health crisis worldwide [12, 13]. According to the World Health Organization (WHO), every year more than 3 million

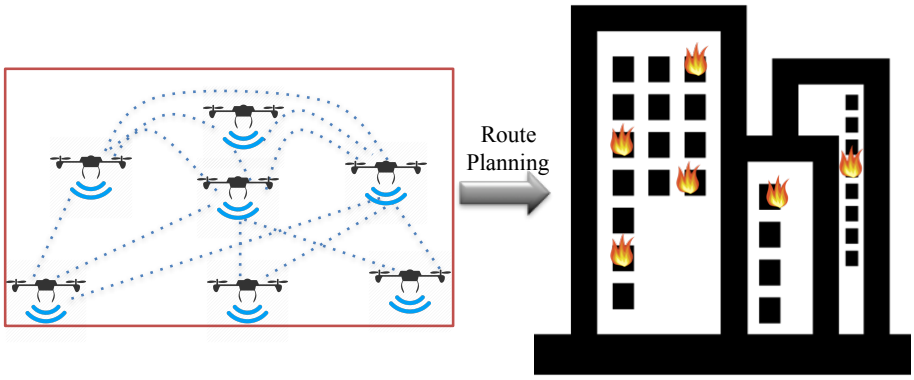


Figure 1.4: This figure shows an example of applying a controllable drone based CPS for urban rescue. In accordance with the needs of domain experts, the CPS is able to autonomously navigate controlled mobile devices (drones) to a set of goal locations to collect situational information, including the location and extent of the fire, location of the survivors, and possible enter and exit routes.

deaths are linked to exposure to outdoor air pollution and 92% of the world's population lives in places where air pollution levels exceed WHO limits [14]. It is beneficial for both residents and city administrations to understand air quality in their immediate environment, with fine-grained temporal-spatial resolution and large coverage [15, 16]. A feasible solution is to install air pollution sensors on cars, especially taxis. When these taxis move around the city, sensors keep on collecting data at different locations and times, which helps to estimate real-time fine-grained city-scale air pollution [17, 18]. With such information, city managers are able to understand pollution levels and pollution sources for further management, planning and development, while residents can arrange their outdoor activity to reduce their exposure risks (e.g. choosing the commute routes or housing with low pollution) [19]. The details of estimating physical field for large scale mobile CPS (vehicle based CPS) can be found in Chapter 4.

1.2.3 Route Planning for Controllable Large Scale Mobile CPS

Controllable drone based CPSs are used in hazard and harsh environments due to their high agility and low cost. Figure 1.4 illustrates how to apply the controllable drone based CPS in urban fire rescue. Before entering the fire scene, it is essential

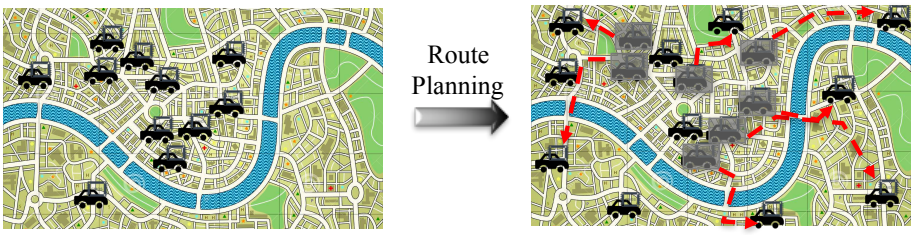


Figure 1.5: This figure shows an example of utilizing a semi-controllable vehicle based CPS for more balanced data collection in crowdsensing. The vehicles are installed with sensors to collect information over time and space. By actuating some vehicles to follow the routes shown in red dotted lines, the vehicle fleet achieves more balanced sensing coverage for data processing in crowdsensing.

for fire fighters to understand the situational information, including the location and extent of the fire, location of the survivors, and possible enter and exit routes. The drone based CPS is able to perform autonomous route planning and navigate controlled micro aerial vehicles (MAVs), to a set of goal locations to collect the key information [20]. The information collected not only helps increase the rescue efficiency but also helps decrease the rescue risk. Route planning for controllable CPSs depends on estimating a system status, i.e. the location of each device. The details of estimating system status and route planning for the controllable large scale mobile CPS (drone based CPS) can be found in Chapter 5.

1.2.4 Route Planning for Semi-controllable Large Scale Mobile CPS

For the large scale mobile vehicle based CPS, it is possible to actuate some of the vehicles if a reward is provided by the system and the driver is willing to accept it. This makes vehicle based CPS semi-controllable and the route planning involves both the system status and the physical field. Figure 1.5 demonstrates how to design route planning to dispatch the semi-controllable vehicle based CPS to sense data with dense coverage and even distribution for crowdsensing. Similar to the application in Figure 1.3, sensors are installed on taxis to collect information over different time and locations. As the left sub-figure shows, the data collected by the vehicle based CPS is uneven since some areas get multiple taxis to collect redundant information, while the other areas are not covered by any taxis. If some of the taxis

can be actuated with low cost, i.e. following the routes shown as red dotted lines, the new distribution of the taxi fleet enables much more balanced sensing coverage, as shown in the right sub-figure. Consequently, the collected data motivates much easier information processing. The route planning of such a semi-controllable CPS depends on the estimation of system status (mobility prediction of each taxi) and physical field (ride request distribution over the city). The details of estimating system status & physical field and route planning for the semi-controllable large scale mobile CPS (vehicle based CPS) can be found in Chapter 6.

1.3 Challenges: Sparse Coverage and Uneven Distribution

The goal of a CPS is to achieve accurate understanding of the physical world according to its application requirements. The collected data from mobile devices in the system should contain a sufficient amount of information for field estimation to achieve the application specific accuracy. In addition, the accuracy of field estimation at different locations and time periods should maintain an (application specific) consistent accuracy level. Since this thesis focuses on physical fields that are continuously changing over space and time, it is important to obtain data that 1) contains at least the minimum amount of information to ensure application specific accuracy and 2) contains a similar amount of information at different spaces and time periods to ensure a consistent accuracy level that does not vary a lot.

Therefore, I define *"good" data collection* in this thesis with two components. 1) **A dense coverage**: the percentage of the overall space and time period being sensed by the mobile devices in the system should exceed a minimum number. This minimum number is calculated according to the application specific accuracy. 2) **An even distribution**: the information entropy of data distribution over space and time should exceed a minimum number. The minimum number is calculated according to the application specific acceptable accuracy variance [21].

However, real systems face the **problems of sparse coverage and uneven distribution**. This is because many mobile devices gather in the same areas to sense redundant information and they also move frequently over time. Consequently, based on the collected data, the field estimation using data-driven methods suffers

from low accuracy due to an insufficient amount of information. In addition, the field estimation using data-driven methods also shows variant accuracy due to fluctuating amounts of sensed information over space and time. Chapter 3 correlates system status from different devices through the same areas and time periods to form a large dataset for model learning and system status estimation. Chapter 4 utilizes the evolution of the physical field over space and time, which indicates the relationship of values at adjacent areas and time periods, to help estimate the physical field.

To improve the coverage and evenness of the data distribution, route planning designs the routes for mobile devices to make sure that they sense data at designated locations and times. Since route planning relies on field estimation, especially system status estimation (e.g. locations of mobile devices), inaccuracy from field estimation deteriorates the performance of route planning. In addition, real-world systems are semi-controllable. Only a fraction of total mobile devices follow the suggested routes from the system. The problems of selecting which mobile devices should be actuated in route planning and how to design routes for the selected mobile devices become challenging. Chapter 5 improves system status estimation of individual devices at their trajectory intersections according to similar radio signatures (physical field), and adaptively plans the routes based on the quality of system status estimation at different times and locations. Chapter 6 collaboratively utilizes vehicle mobility (system status)) and ride requests (physical field) at adjacent areas and time periods to calculate an optimal solution for route planning.

1.4 Research Statement

The thesis presents a spatiotemporal relationship aided framework for large scale mobile CPSs to address the challenge of sparse coverage and uneven distribution of data collection.

By utilizing the spatiotemporal relationships of physical field and system status, which are discussed in Section 2, the *spatiotemporal relationship analysis layer* improves the performance of major system tasks: field estimation (physical field and system status) and route planning on three types of large scale mobile CPSs.

Three different types of spatiotemporal relationships are introduced in the *spatiotemporal relationship analysis layer*. 1) The *spatiotemporal relationship of physical*

field (ST-PHY): the evolution of the physical field over space and time, determined by the law of physics, indicates the relationship of values at adjacent areas and time periods, such as air pollution levels in a city across a day. 2) The *spatiotemporal relationship of system status (ST-SYS)*: system status changes continuously over time and space for the same device and the system status of the same type of devices may have similar values at the same areas and time periods. 3) The *spatiotemporal relationship connecting physical field and system status (ST-PHY-SYS)*: the physical field and system status are interrelated through time and space because they both react to the physical world.

Models and algorithms are designed based on the spatiotemporal relationship aided framework for field estimation and route planning. Implementations of uncontrollable, controllable and semi-controllable large scale mobile CPSs are presented to show how this framework works with different types of CPSs. Both real deployed testbed experiments and extensive simulations with real world collected data are adopted to validate the system design. As a part of the evaluation for uncontrolled to controlled motion aspects of our system, air pollution sensors are deployed on the taxi-based testbed to collect data in the city of Shenzhen for 2 years in collaboration with Tsinghua University. In addition, a swarm of 8 micro aerial vehicles are deployed in an indoor environment for autonomous navigation. The results show incorporating the *spatiotemporal relationship analysis layer* can achieve 2.1× and 6× error reduction on physical field and system status estimation and 3× improvement on route planning. This illustrates the potential of the *spatiotemporal relationship analysis layer* to make large scale mobile CPSs more intelligent on field estimation and route planning.

1.5 Organization

The remainder of the thesis is organized as follows. We first introduce the spatiotemporal relationship aided framework in Chapter 2. Then we discuss how the *spatiotemporal relationship analysis layer* improves estimating system status and physical field for large scale mobile CPSs in Chapter 3 and Chapter 4 respectively. After that, in Chapter 5, I introduce how the *spatiotemporal relationship analysis layer* helps route planning for the controllable large scale mobile CPS, which involves

estimating system status. After that, I illustrate how the *spatiotemporal relationship analysis layer* assists route planning for the semi-controllable large scale mobile CPS in Chapter 6. The route planning depends on estimation of both system status and physical field. Finally, we conclude the thesis in Chapter 7.

Chapter 2

Solution Overview: Spatiotemporal Relationship Aided Framework for Large Scale Mobile Cyber Physical Systems

To address the challenges of sparse coverage and uneven distribution, as described in Section 1.3, we focus on how physical field and system status change across space and time to improve the performance of field estimation and route planning. In this thesis, the *spatiotemporal relationship* is defined as the relationship between values at adjacent spatial and temporal points, which consists of three different types. 1) The *spatiotemporal relationship of physical field (ST-PHY)*: the evolution of the physical field over space and time, determined by the laws of physics, indicates the relationship of values at adjacent areas and time periods, such as air pollution levels in a city across a day. 2) The *spatiotemporal relationship of system status (ST-SYS)*: system status changes continuously over time and space for the same device and the system status of the same type of devices may have similar values at the same areas and time periods. 3) The **spatiotemporal relationship connecting physical field and system status (ST-PHY-SYS)**: the physical field and system status are interrelated through time and space because they both react to the physical world.

The thesis presents a spatiotemporal relationship aided framework for large scale

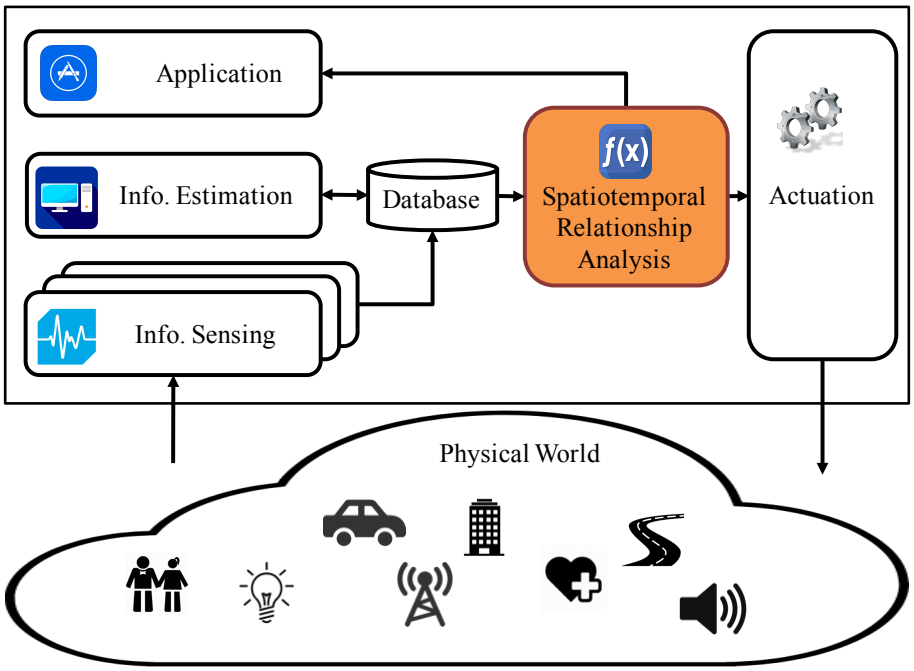


Figure 2.1: The figure shows the spatiotemporal relationship-aided CPS framework. The newly designed *spatiotemporal relationship analysis layer* utilizes the 3 types of spatiotemporal relationships to improve the performance of field estimation and route planning, which offers more informative and accurate understanding of the physical world for different applications.

mobile CPS, which incorporates a new *spatiotemporal relationship analysis layer*, as shown in Figure 2.1. Along with the newly designed *spatiotemporal relationship analysis layer*, the spatiotemporal relationship-aided large scale mobile CPS is composed of one *database*, two cyber components dealing with cyber information: the *information estimation layer* and the *application layer*, and two physical components interacting with physical processes: the *information sensing layer* and the *device actuation layer*.

- The *information sensing layer* collects data from the physical world with the help of various sensors, such as temperature, moisture, sound, air pollution, video, etc. The system deploys these sensors on a large number of mobile devices, which helps acquire information over the temporal and spatial domain. The collected data is stored in the *database* for further processing in the *information estimation layer*.
- The *information estimation layer* preforms two important **field estimation**

tasks with the collected data stored in the *database*: **estimating physical field and system status**. Estimating the physical field gives the system more information on the physical world,, which is used for further decision making in the *application layer* and route planning in the *device actuation layer*. Estimating system status enables the system to infer important information about the system and maintain a good operational status, which helps route planning in the *device actuation layer*. The estimated results are stored back to the *database*.

- The *device actuation layer* actuates the mobile devices in the system according to the estimated physical field and system status from the *information estimation layer*. In this thesis, I focus on **route planning**, which dispatches mobile devices to get better data collection. On the one hand, the field estimation from the *information estimation layer* affects how the *device actuation layer* dispatches large scale mobile devices. On the other hand, the dispatching results from the *device actuation layer* affects how well the *information sensing layer* can sense the physical world, thus affecting the field estimation in the *information estimation layer*.
- The *application layer* utilizes the derived estimation stored in the *database* for various high level purposes. The *application layer* holds the interface between CPSs and human beings, which helps people better monitor and control the physical world.
- The *spatiotemporal relationship analysis layer* first derives information from either the raw sensed data by the *information sensing layer* or the estimated results from the *information estimation layer*, both of which are stored in the *database*. Then, the *spatiotemporal relationship analysis layer* utilizes the spatiotemporal relationships of physical field and system status to further process the information to improve the performance of field estimation or route planning.

For the large scale mobile CPS, estimating the system status is essential to ensure the system maintains a stable operation status. In recent years, human beings have been regarded as one of the most important components in the loop of CPS [9]. Therefore, as part of the system, human behavior is regarded as a system status. The challenge here is two-fold. 1) Human behavior is multifaceted and involves many

factors of information. 2) The amount of data a single device can collect is limited for model learning. To address these two challenges, the spatiotemporal relationship of system status (ST-SYS), i.e. the similarity of human behaviors at same or close temporal and spatial points, is utilized. The details can be found in Chapter 3.

In addition, estimating the physical field is another essential task of field estimation, which helps the system to understand the physical world based on the sensing data. However, purely applying the traditional data-driven methods, especially machine learning methods, to estimate leads to inaccurate and unstable estimation due to sparse coverage and uneven distribution. To address the challenge, the evolution of physical fields over space and time (ST-PHY), determined by the law of physics and described as a physics guided model, which indicates the relationship of values at adjacent areas and time periods, is adopted to help estimate the physical field. The spatiotemporal relationship between the physical field and the sensed data is used to infer the values of the physical field when and where no data is sensed, offering extra information for estimation. Knowing the (temporal and spatial) correlation between nearby data points also prevents the system from being biased by inaccurate data-driven estimation. Moreover, to ensure stable and accurate physical field estimation, features such as the amount of data collection and the estimation quality at different areas and times (ST-SYS) are used to adaptively switch between the physics guided model and the data driven model. The details can be found in Chapter 4.

The controllable system is able to dispatch mobile devices to key locations in order to to collect more informative data to understand the physical process. Route planning depends on the results of system status estimation, whose accuracy may be low due to the limited capabilities of each individual device in sensing, computing, and communication. In addition, the accuracy of the estimation on system status may vary over time. To address the challenges, the *spatiotemporal relationship analysis layer* improves the system status estimation of individual nodes at their trajectory intersections according to similar radio signatures (ST-PHY-SYS). Based on the quality of system status estimation at different times and locations (ST-SYS), the system adaptively plans the routes. The details can be found in Chapter 5.

Compared to the controllable large scale mobile CPS, semi-controllable ones have more challenges in route planning, since their route planning involves estimating both system status and physical field. Along with the challenges of controllable large scale

mobile CPSs, semi-controllable systems have to face the problem of selecting which mobile devices should be actuated in route planning and how to design routes for the selected mobile devices. When this is applied to cars, the system predicts the vehicles' mobility to help the system to choose the "correct" vehicles to actuate (ST-PHY), and the ride requests over the city to help lower the incentive cost (ST-SYS). The system combines information of mobility of all the vehicles and the ride requests over the city to calculate an optimal solution for route planning (ST-PHY-SYS). The details can be found in Chapter 6.

Chapter 3

Estimating System Status for Large Scale Mobile CPS

This chapter introduces how to utilize spatiotemporal relationships to address the challenge of sparse coverage and uneven distribution in system status estimation (human App usage behavior prediction) as shown in Chapter 1. Estimating system status is essential to ensure the system maintains a stable operation status. In recent years, human beings have been regarded as an important components in the loop of CPS [9]. I focus on estimating human behavior (system status) pattern, where the system has no control over the route of human beings in this chapter. The controllable and semi-controllable system will be discussed in Chapter 5 and 6 respectively.

Compared to traditional cyber or physical systems, the status of human beings is more complex. Consequently, I need information from multiple domains to estimate the system status, i.e. the human behavior. Therefore, according to Chapter 2, the *spatiotemporal relationship analysis layer* figures out the interaction of these influential factors through their spatiotemporal relationships (ST-SYS). In addition, the amount of data collection from any single device is limited, while the behavior patterns of different individuals may vary. Different human beings in the same place and time show correlated data. The correlated dataset from all human beings in the system is used for model learning and the data from each person is used for personalized prediction. This ensures sufficient information for model learning and keeps personalization for different individuals.

In this chapter, multi-domain information (time, location, App usage, and App type) is correlated through their spatiotemporal relationship (ST-SYS) to learn human App usage patterns. The system also adopts App usage history from all users to ensure sufficient data for model learning and App usage history from each individual to generate the user profile for personalized prediction.

I first explain the problem in Section 3.1. Then I describe the data collection and problem definition in Section 3.2. After that, I introduce the high level system design in Section 3.3 and key algorithm design in Section 3.4. The evaluation is discussed in Section 3.5. Finally, I discuss the related work in Section 3.6 and conclude the section in Section 3.7.

3.1 Problem Overview

The smart device market has been showing continuous and rapid growth in the last decade. Mobile phones alone show a projection of 2.53 billion worldwide users by 2020 [22]. These smart devices are mainly used with mobile Apps, which have project revenues of around 189 billion US dollars by the year 2020 [23]. Currently, around 2.8 million and 2.2 million Apps have been developed and made available in Google Play and Apple App Store respectively.

With this explosive growth of the mobile App market, accurately predicting users' App usage is essential for carriers, consumers and advertisers. Because the projection of mobile data usage by 2021 is 48 exabytes per month, carriers will need more accurate and dynamic bandwidth allocation schemes to increase bandwidth utilization efficiency [11]. Predicting consumers' App usage at specific times and locations helps carriers understand consumers' bandwidth needs more precisely for smart bandwidth allocation. For consumers, App usage prediction information not only helps better battery life prediction and management, but also accelerates App launching. In addition, advertisers can recommend advertisements to the Apps that users will most likely use given location and time.

Prior works have attempted to predict mobile App usage [24, 25, 26]. Church et al. [27] summarized the challenges for mobile phone usage learning and analysis and describe a series of studies and applications on mobile phone usage, including App recommendation [28], launcher prediction [29], and battery management [30].

Various prediction algorithms have been explored to achieve that goal. Kostakos et al. [31] applied a Markov state transition model to predict the next screen event. Xu et al. [32] proposed a multi-faceted approach to predict App usage. The study focused on a small-scale dataset, posing a key challenge to understand and predict App usage behavior over a large user population. Shin et al. [29] predicted the App usage based on a personalized Naïve Bayes model for each user profiled from the usage data from their phones. Since their prediction is based on individual historical data and contextual information, it is limited by what a user has already experienced. In addition, various algorithms and information types have been explored for prediction and recommendation in different domains. Context aware recommendation is achieved by using information about location, time, and activity [33, 34, 35, 36, 37]. Context aware collaborative filtering and recurrent neural network are proposed for activity recommendation, App recommendation, and location prediction. In addition, Berkel et al. [38] looked into a different aspect of smartphone usage by classifying usage gaps to identify the usage session. They also use user profile information to achieve personalized recommendations for news, blogs, Apps, and e-commerce items [39, 40, 41, 42, 43]. Until now, no research focuses on personalized App usage prediction on a large scale population using both temporal and spatial information.

The goal of this chapter is to consistently predict which App will be used given a user's time and location over a large scale user population. Despite the related work mentioned before, challenges remain. *i.* Mobile App usage behavior is complicated. What are the key factors that affect the prediction? It is also difficult to derive the importance of these factors. *ii.* A user's App usage preference is decided by multiple factors and time-variance, which is difficult to describe. *iii.* A user's data is sparse on the spatial domain. One user only covers a limited number of locations. Prediction is difficult if the user appears at a new location. Accurate prediction is impossible without addressing these challenges.

I present *CAP*, a context-aware App-usage prediction system that handles the aforementioned challenges. To address challenge *i.*, I design a heterogeneous graph embedding algorithm, which utilizes the the spatiotemporal relationship (ST-SYS) to map time, location, App, and App type into one common latent space. The embedding catches the relationships between App-location, App-time, and App-App type. To address challenge *ii.*, I design user profiles with users' past App usage

and trajectory, which are affected by a time decay factor, to describe the individual dynamic preference. Finally, I adopt the history data of all users to construct a heterogeneous graph for an individual user to generate personal preference, which addresses challenge *iii.* and keeps my prediction personalized. My contributions are listed as follows:

- I am the first to investigate the context-aware App-usage prediction problem over a large user population. I consider context information (time & location), attribute information (App & App type) and dynamic user preference.
- I find that the relationships between App-location, App-time, and App-App type are essential to prediction and propose a heterogeneous graph embedding algorithm to map them into one common comparable latent space. I propose a user profile with personal App usage & trajectory history affected by a time decay factor to achieve a personalized prediction. I extract both the common attribution of all users and individual user dynamic preferences to ensure sufficient training data without losing personalization.
- I evaluate my system through a large-scale real-world dataset, which includes more than 6 million mobile App usage records from 1788 individual users. *CAP* demonstrates a significant improvement in the prediction accuracy compared to baselines.

3.2 Dataset and Problem Description

This subsection introduces the preliminaries of the mobile App usage prediction. I first introduce the App usage record dataset used for prediction, including the data collection and processing. Then, I formally define the prediction problem.

3.2.1 Data Collection and Processing

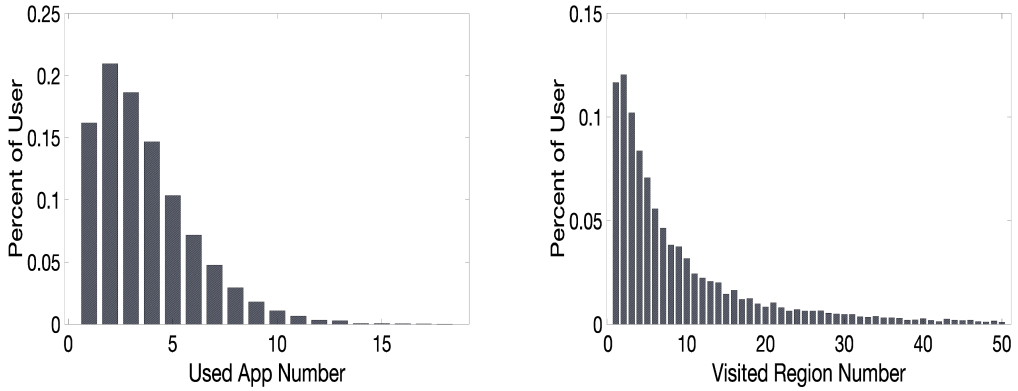
The App usage record dataset is collected with Deep Packet Inspection (DPI) appliances [44], through China TeleCom, a major cellular network operator in China [45]. It records the spatio-temporal information of mobile subscribers when they access cellular network for App usage. Thus, the recorded locations are at the granularity of cellular base station. In the dataset, each entry contains an

anonymized user identification, timestamps of HTTP request or response, the length of the packet, the domain visited and the user-agent field. The data is collected in Shanghai, one of the largest cities in China.

I extract the information of what App is used for network requests. The HTTP header captured by my DPI uses various fields to identify the Apps as they communicate with their host servers or third party services. The hosting servers need to distinguish between different Apps in order to provide appropriate content. I therefore identify the App making a network request by inspecting those HTTP header identifiers. I utilize a systematic framework for classifying network traffic generated by mobile Apps: SAMPLES [46]. It uses constructs of conjunctive rules against the App identifier found in a snippet of the HTTP header. The framework operates in an automated fashion through a supervised methodology over a set of labeled data streams. It has been shown to identify over 90% of these Apps with 99% accuracy on average [46]. In order to obtain the labeled dataset, I crawled the 2000 most popular Apps across Apple App Store (iOS Apps) and Google Play (Android Apps) and applies SAMPLES to generate conjunctive rules to match each App's network traffic. I manually verify the correctness of the matched Apps, which achieves about 97% accuracy. In addition, I first extract all cellular network connections for each App. Then, to avoid the repetitive count of App sessions, I adopt density-based spatial clustering of applications with noise (DBSCAN) to cluster the App sessions. Each cluster is regarded as one session [47].

I also classify App types for attribute information. The App type indicates an App's functionality and attribute. Each App is categorized into at least one of the nineteen App types, including game, video, news, social, E-shopping, finance, real estate, tourism, daily service, education, therapy, baby caring, taxi, vehicle relevance, music, map, reading, vogue, and office. I manually assign each App name an App ID and each App type an App type ID for easy successive processing.

Dataset anonymization: It is worth pointing out that privacy issues of this dataset are carefully considered, and I take measures to protect the privacy of these mobile users. The App usage record dataset does not contain any personally identifiable information. The user identities have been anonymized as a bit string and do not contain any user meta-data. All the researchers are regulated by a strict non-disclosure agreement and the dataset is located in a secure offline server. Relevant privacy laws and institutional policies of both P.R. China and the United States are



(a) Statistics of mobile App usage number in 7 days

(b) Statistics of mobile App usage region number in 7 days

Figure 3.1: This figure shows statistics of mobile App usage number and region number in 7 days. Most users use fewer than 10 mobile Apps in in fewer than 20 regions. More than 80% users use more than 1 mobile Apps.

strictly followed.

The App usage record dataset is able to support our research in terms of the following aspects. First, each record reflects who uses what App at what location and time. Although this dataset misses the App usage activities that are connected through WiFi, these App usage activities lose the location information, which does not reflect a user's temporal-spatial App usage pattern. In addition, most App usage activities involve network connections. Second, the dataset covers one week and the whole metropolitan area of the city. This ensures a large temporal and spatial information range, which shows how users' App usage patterns are related to time and location. For example, users tend to use food delivery Apps during lunch in the office area. Third, the dataset includes 6 million App usage records from 1788 million users, which guarantees that I have enough data for my context-aware App usage prediction research. Fourth,including the 2000 most popular Apps from the Apple App Store (iOS Apps) and Google play (Android Apps) ensures that all the most commonly used Apps can be predicted. Finally, a single cellular base station covers multiple regions of different kinds, such as shopping malls, schools, offices etc., which indicate both semantics and geographical information. An individual user usually goes to one typical region given one specific cellular base station. Therefore, adopting cellular base station to infer individual preference is reasonable.

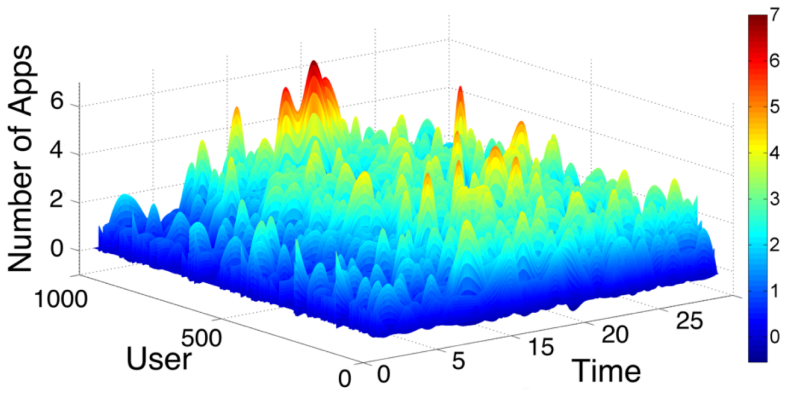
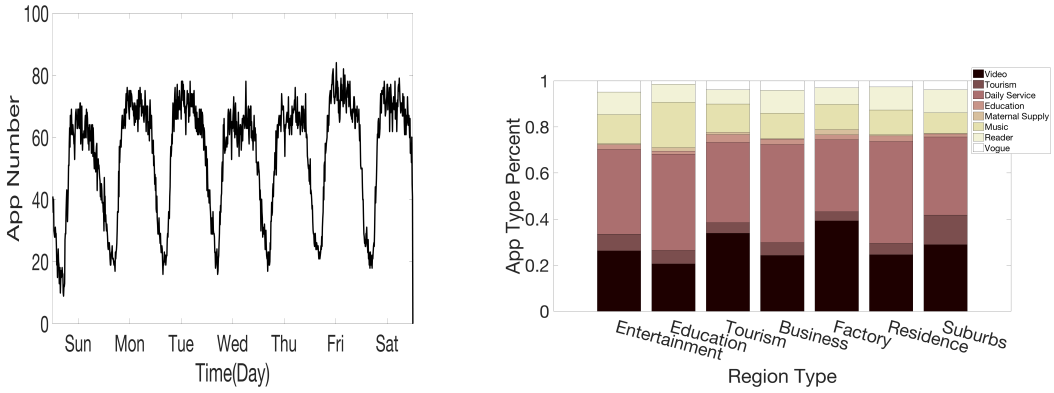


Figure 3.2: Mobile App usage number of 1000 users at different time.

Dataset Characteristics: Figure 3.1 shows the statistics of number of mobile App usage and regions covered by cellular base stations in a week. Most users use fewer than 10 mobile Apps in a week and they use these mobile Apps in fewer than 20 regions. This shows that people focus on a limited number of mobile Apps and stay in limited areas. However, more than 80% users use more than one mobile App. Figure 3.2 shows mobile App usage by 1000 users at different times. Figure 3.3(a) shows total mobile App usage by 1000 users checked every 10 minutes in one week. Mobile App usage is highly correlated with the human activity pattern, in which people use more Apps during daytime and fewer Apps at night. In addition, different users show variant mobile App usage numbers and peak times. Figure 3.3(b) shows that Apps of different types have different usage patterns in different location types. All these observations illustrate the potential to predict users' mobile App usage pattern with context information (location and time). In addition, mobile App usage is time-variant, user-variant and location-variant. As a result, it is necessary to figure out a novel solution to do context-aware mobile App usage prediction.

3.2.2 Problem Description

In order to predict context-aware App usage patterns, i.e. what App a user will use given the time and location, I first define the problem as follows. Let $\Re$ be a corpus of mobile user App usage records. I apply U, C, T, A, P to represent user identity, location, time, App and App type respectively. I use a subscript to denote the record id. For the kth record, it is a tuple $< U_k, C_k, T_k, A_k, P_k >$, where U_k and



(a) Temporal distribution of total mobile App usage number with 10 minutes resolution

(b) Apps of different types have different usage patterns in different location types

Figure 3.3: This figure shows (a) total number of App used every 10 minutes in one week, and (b) the usage of Apps at different types of locations.

C_k are user ID and cellular tower ID, while T_k, A_k and P_k are time, App ID and App type ID. All records are sorted chronologically, with smaller k meaning earlier time. I aim to correlate the mobile App usage pattern to time and location with $\Re$, which consists of large amounts of App usage records. Given five different interconnected factors, an effective and fast model is needed to accurately capture the cross-modal correlation among C, T, A, P for each user.

Based on the App usage record dataset $\Re$, given a querying user u ($U_k = u$) with the context of time T_k and connected cellular towers C_k (query $q = (u, T_k, C_k)$), my system predicts top N mobile Apps the user u will probably use. The prediction is based on the mobile App usage history of all users, i.e. all records in $\Re$ with an earlier time than T_k. The system outputs the top N most likely mobile Apps user u will use. Instead of only applying user u's own historical data, the system adopts the history data of all users due to two reasons. Firstly, user u may not have large amounts of historical data to figure out the mobile App usage pattern. Secondly, the history data from only one user only contains a limited number of mobile Apps and locations. This leads to a wrong prediction if a user appears at a new location or uses a new mobile App.

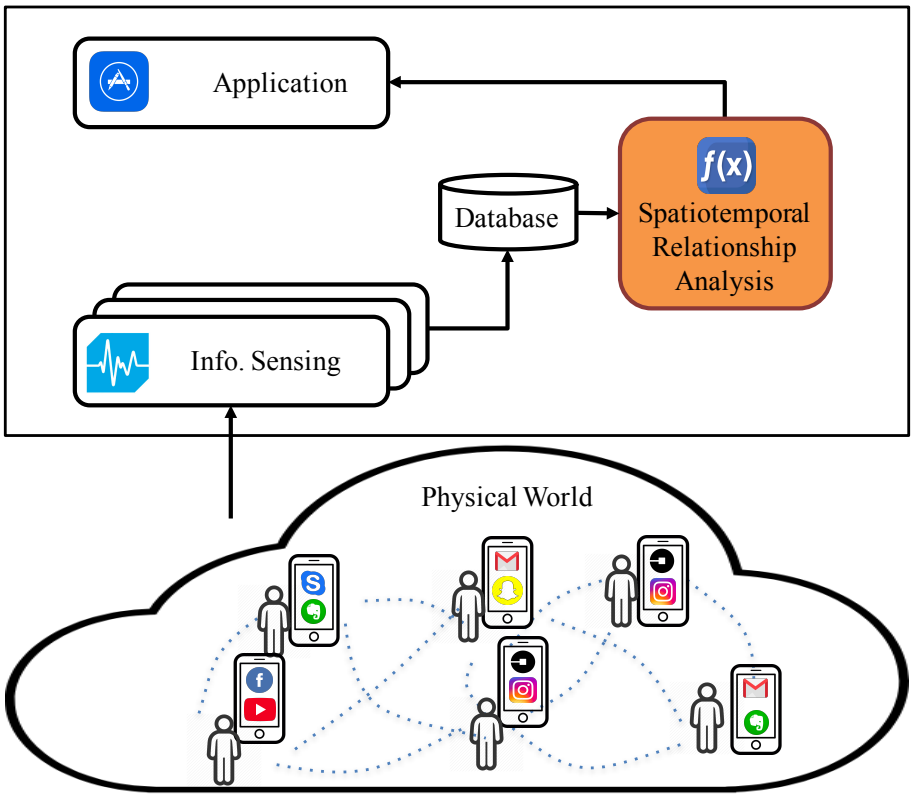


Figure 3.4: The figure shows the system architecture of spatiotemporal relationship aided large scale mobile phone based CPS for system status estimation (personalized context-aware App usage prediction).

3.3 System Design

This subsection introduces system design intuition for the large scale mobile phone based CPS to address the challenges described in Section 3.1. Figure 3.4 shows the high-level architecture of the large scale mobile phone based CPS. Users' data is collected through the *information sensing layer* and stored in the *database*. The *spatiotemporal relationship analysis layer* gets users' data from the *database* to learn the users' usage pattern and predict the App usage given a user's location and time.

The detailed architecture of the *spatiotemporal relationship analysis layer* is designed as Figure 3.5. First, all users' history data is sent to the *Representation Learning* module after the *Pre-Processing* module, which ensures a large amount and high heterogeneity of information for model training. According to co-occurrence detection and attribution detection, this module learns a heterogeneous graph

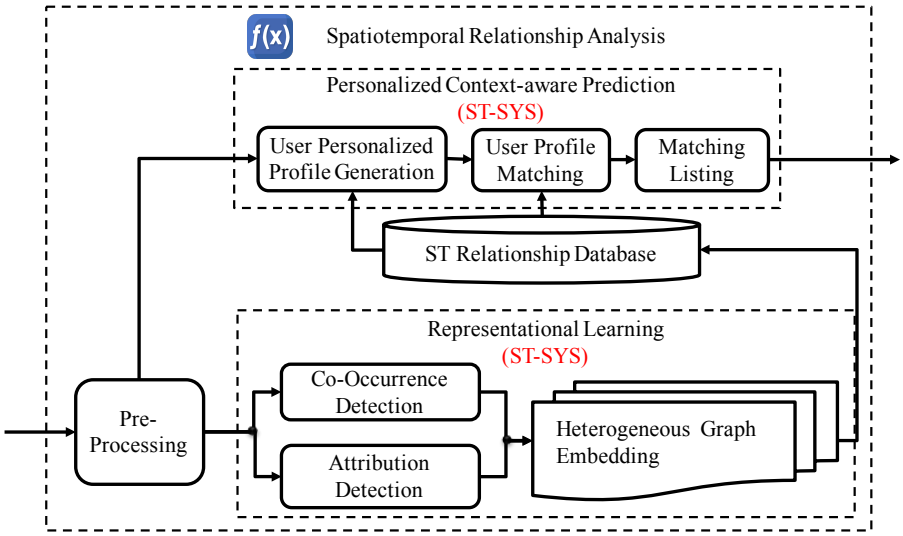


Figure 3.5: This figure shows the architecture of the *spatiotemporal relationship analysis layer*, which utilizes the spatiotemporal relationship (ST-SYS) of different influential factors from different human beings to help the large scale mobile CPS (mobile phone based CPS) for system status estimation (personalized App usage prediction).

embedding model to compare information from different dimensions. The learned model is stored in the *spatiotemporal relationship database*. The *Context-aware App-usage Prediction* module generates a user profile for each user based on the trained model from the *spatiotemporal relationship database*, which is used for personalized prediction.

Pre-Processing module prepares clean data for successive processing. As shown in Figure 3.5, each App usage record includes an anonymous user ID U, a connected cellular base station ID C, a time stamp T and an App ID A, which captures the information of user identity, location, time and App usage. The module first removes conflicting and redundant App usage records by checking the time stamp and cellular base station ID. Then, I remove the records of the Apps used at all locations and times. These Apps include wechat (Chinese WhatsApp), weibo (Chinese twitter) etc. I remove them based on the following reasons: 1) A user can use them at any location and any time, which is difficult to predict. 2) Adding records of these Apps does not contain much useful information. 3) Other predictable Apps' usage patterns will be overwhelmed by a large amount of these unpredictable App records. In addition, the App attribute dataset includes the information of App ID A and App type ID P, which capture the common attribution of Apps. Each App usage record is associated

with one App attribute record by matching the same App ID.

Representation Learning module gathers information of different dimensions from all users' historical data for training. This ensures both a large amount and high heterogeneity of information for training. Information from different dimensions is mapped into a latent space based on two kinds of relationships *co-occurrence* and *attribution*. A heterogeneous graph-based learning method is designed to make information from different dimensions comparable. The module outputs the trained model and the embedded vectors in the latent space, both of which are stored in the *spatiotemporal relationship database*. The details can be found in Section 3.4.1.

Context-aware App-usage Prediction module, based on the trained model stored in the *spatiotemporal relationship database*, first adopts each user's history data to generate user profile to describe dynamic user preference in the *user personalized profile generation* sub-module. The profile is based on two basic elements: a user's mobile App usage history and his/her past trajectory, both of which are affected by a time decay factor. The personalized user profiles are expressed with vectors in the latent space. This module then predicts what App a user will use given the location C and time T. After that, the *user profile matching* sub-module matches the personalized user profile vectors with history record vectors in the latent space from *spatiotemporal relationship database*. Based on the matching scores, this module outputs the top N predicted mobile Apps. The details can be founded in Section 3.4.2.

3.4 Algorithm Design

In order to make the information of time, location, App, and App type comparable and figure out the importance and intersections of these factors, I adopt representational learning technology to map the information into a common latent space. To be more specific, I designed a graph-based embedding method, which provides the following benefits. First, it preserves the direct occurrence interactions between factors, such as time, location and App in the same App usage record. Second, it also keeps indirect attribution interactions between records through Apps belonging to the same App type. Finally, it lowers the dimension needed to represent these factors by extracting two kinds of interactions. Take the App representation as an example. Traditional one hot representation requires a vector of 2000 dimensions to represent

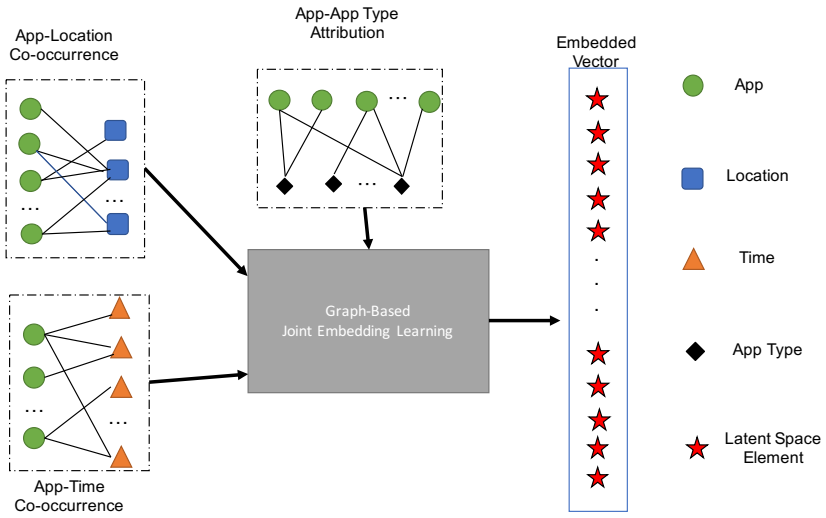


Figure 3.6: This figure shows how I encode the *co-occurrence* and *attribution* relationships and adopt graph-based joint embedding learning to map these relationships into one latent space.

2000 Apps [48]. In contrast to this method, embedding first extracts direct and indirect structures between the 2000 Apps given the users' history App usage records and the App attribute dataset. Then, the 2000 Apps are mapped into a latent space of much lower dimension based on the learned interactions. In my case, only 20 dimensions are needed.

3.4.1 Representational Learning

High-quality embedding requires preserving both direct and indirect factor interactions, which are defined as *co-occurrence* and *attribution* respectively. The *co-occurrence* relationship captures the direct interaction between user, time, location and App, i.e. who uses what App at what location and time. It preserves the information where and when an App is used. The *co-occurrence* relationship happens when two units show up in the same record. For example, *Pre-Processed Data* module outputs a record with a time unit (e.g. 6:50 PM), a location unit (e.g. cellular base station ID 272368) and an App (e.g. App ID 223). Two *co-occurrence* relationships reflect direct spatial and temporal usage correlation: App-location and App-time. The *attribution* reflects the indirect semantic interactions of these factors, i.e. correlating time, locations and Apps from different App usage records based on

similar App attribute, which is expressed by App type information. The *attribution* relationship comes from the assumption that Apps belonging to the same type share similar attribution and users tend to use Apps belonging to the same type.

I use bipartite graphs to encode the *co-occurrence* and *attribution* relationships for further embedding learning, as shown in Figure 3.6. The graph has four different node types which correlate to four factors, App, location, time and App type respectively. The edges are constructed based on *co-occurrence* and *attribution* relationships.

App-Location Graph captures the spatial attribution of mobile App usage and is denoted as $G_{AC} = (A \cup C, \varepsilon_{AC})$, where A and C represent the mobile App Id and connected cellular base station ID. The edge ε_{AC} connects mobile App nodes and cellular base station nodes. Edge weights w_{AC} are set to the normalized *co-occurrence* counts.

App-Time Graph captures the temporal attribution of mobile App usage and is denoted as $G_{AT} = (A \cup T, \varepsilon_{AT})$, where A and T represent the mobile App Id and time. The edge ε_{AT} connects mobile App nodes and time nodes. Edge weights w_{AT} are set to the normalized *co-occurrence* counts.

App-App Type Graph captures the interactions of Apps with similar functions, i.e. belonging to the same App type, which is denoted as $G_{AP} = (A \cup P, \varepsilon_{AP})$. A and P represent the mobile App Id and mobile App type Id. If a mobile App $A_i \in P_j$, there is an edge $\varepsilon_{A_iP_j}$ connecting mobile App node A_i and mobile App type node P_j. In order to reflect the importance of different Apps, the TF-IDF is applied to derive edge weights $w_{A_iP_j}$ [49]. I treat Apps as a bag of words and App types as documents. I input counts of all Apps in all App types. Let $n_{i,j}$ be the count of the App i in the App type j, then I calculate the term frequency $TF_{i,j}$ of the App i in the App type j by $TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$. Let $|P|$ be the total number of App types, and p_j represent the App type j, the IDF for App i will be $IDF_i = \log \frac{|P|}{1+|p_j \in P: a_i \in p_j|}$. The final $TF\text{-}IDF$ value for the App i in the App type j is calculated as $TF\text{-}IDF_{i,j} = TF_{i,j} \times IDF_i$, which is used as the weight of the edge in App-App type graph.

The three graphs above capture the temporal, spatial and semantic effect of users' App usage respectively. Take the App-location graph as an example for interpretation. If a mobile App A_i is often used in location C_j, the edge weight w_{AC} is large. As a result, given a target user u at location C_j, he/she will most likely use mobile App A_i. These three graphs are embedded into a shared low dimensional latent space

R^d, whose dimension is d. In the latent space, App, time, location, App type are represented as $\vec{a}$, $\vec{c}$, $\vec{t}$ and $\vec{p}$.

The goal of graph embedding learning is to represent the graph nodes in lower dimension while preserving the structure. In addition, due to the heterogeneity of the three graphs, I need to rank them in order of importance. I first model the emission probability distribution of each node according to the latent embeddings and then minimize the distance between the distributions and really observed distributions. The joint training described in Algorithm 1 is designed to iteratively optimize the overall loss function of three graphs.

Given a bipartite graph $G_{XY} = (X \cup Y, \varepsilon_{XY})$, where X and Y are two sets of nodes representing different information types and ε_{XY} is the set of edges connecting them, the likelihood of generating node j given node i is defined as

$$p(j|i) = \frac{\exp(-u_j^T \cdot v_i)}{\sum_{k \in X} \exp(-u_k^T \cdot v_i)}, \tag{3.1}$$

where u_j and v_i are embedded vectors of node j in Y and i in X respectively. Each node i has two different embedding vectors according to their function: v_i when i acts as given node and u_i when i acts as emitted node. The true observed distribution of generating node j given node i is defined as

$$\hat{p}(j|i) = \frac{w_{ij}}{d_i}, \tag{3.2}$$

where w_{ij} is the edge weight, $d_i = \sum_{k \in X} w_{ik}$ and node i belongs to type X.

Before minimizing the distance between the embedding-based distributions and really observed distributions, I define the loss function for the graph G_{XY} as

$$L_{XY} = \sum_{i \in X} d_i KL(\hat{p}(\cdot|i)||p(\cdot|i)) + \sum_{j \in Y} d_j KL(\hat{p}(\cdot|j)||p(\cdot|j)), \tag{3.3}$$

where node i and j belong to type X and Y respectively. $KL()$ is Kullback-Leibler divergence [50]. Since there are three different graphs in my system, the overall loss function is derived as

$$L = L_{AT} + L_{AC} + L_{AP}, \tag{3.4}$$

Algorithm 1: Joint training heterogeneous graph

Input: G_{AC}, G_{AT}, G_{AP}, number of samples M, number of negative samples L
Output: App embedded vector $\vec{a}$, location embedded vector $\vec{c}$, time embedded vector $\vec{t}$ and App type embedded vector $\vec{p}$
while $it \leq M$ **do**
    sample an edge from ε_{AC} and draw L negative edges, update App and location embedded vectors;
    sample an edge from ε_{AT} and draw L negative edges, update App and time embedded vectors;
    sample an edge from ε_{AP} and draw L negative edges, update App and App type embedded vectors;
end

where L_{AT}, L_{AC} and L_{AP} represent loss function of App-location, App-time, and App-App type graph.

Keep in mind that it is computationally expensive to optimize loss Eq. function 3.3 since calculating the likelihood $p(j|i)$ requires sum over the entire set of node in X. To address this problem, a negative sampling approach is adopted, in which I sample multiple negative edges. The sampling is based on noisy distribution for each edge [51]. Then, an asynchronous stochastic gradient (ASGD) algorithm is used for optimization [52]. Since three graphs in my system are heterogeneous and cannot be optimized simultaneously by merging all the edges together, I adopt a joint training algorithm, as shown in Algorithm 1, to iteratively optimize the overall loss function Eq. 3.4 to get d dimensional embedded vectors for App, time, location and App type in the common latent space.

3.4.2 Context-aware App-usage Prediction

To achieve context-aware App-usage prediction, the system needs not only to embed information from different spaces into one latent space, but also to generate a dynamic user profile for each user to describe his/her dynamic preference. The profile should also be mapped into the same latent space as App, location, time and App type for prediction.

User profile generation: I generate the user profile based on two observations: 1) A user's current App usage is related to his/her past App usage and the visited

locations with high probability. 2) Recent App usage and visited locations should play more important roles than old ones. Therefore, given a time τ and a user u, I first extract his/her App usage records before τ, i.e. all records whose $U = u$ and $T < \tau$. These records form a sub-set $\Re^u_\tau \subset \Re$, which consists of records of (u, t_i, c_i, a_i). Then I define the profile of the user u at time τ as:

$$\vec{u}_\tau = \beta \sum_{(u,\vec{c}_i,\tau_i)\in\Re^u_\tau} e^{-(\tau-\tau_i)}\vec{c}_i + (1-\beta) \sum_{(u,\vec{a}_i,\tau_i)\in\Re^u_\tau} e^{-(\tau-\tau_i)}\vec{a}_i, \tag{3.5}$$

where $\vec{a}_i$ and $\vec{c}_i$ are the embedded vectors of the App a_i and the location c_i in $\Re^u_\tau$. $e^{-(\tau-\tau_i)}$ is a time decay factor indicating that older data has less influence. $\beta \in [0, 1]$ is the coefficient to tune the importance of App usage history and trajectory history.

Given a query $q = (u, \tau, c)$, I first obtain the user u's profile at time τ, expressed as $\vec{u}_\tau$. Then I compute scores of different possible mobile Apps a_j as:

$$S(q, a) = \vec{u}_\tau \cdot \vec{a}_j, \tag{3.6}$$

where $\vec{a}_j$ is the embedded vector of mobile App a_j and $\vec{u}_\tau$ is the user profile calculated as Eq. 3.5. The above score not only captures the App usage preference, but also captures the user history trajectory and considers the time decay effect. Based on the score ranking of different mobile Apps, the system outputs N most possible mobile App predictions.

3.5 Evaluation

In this section, I evaluate my system with real world-collected mobile App usage data. I first introduce how I set up evaluation in Section 3.5.1. Then I analyze the system performance in Section 3.5.2. The evaluation focuses on 1) comparing the performance of my system with four different baselines and 2) investigating the influence of key parameters in the system.

3.5.1 Evaluation Setup

Data Pre-Processing: The dataset used for evaluation contains more than 6 million mobile App usage logs from 1788 users in one week, April 19, 2016 - April

26, 2016. These logs record who uses what App at what time and locations. Each record includes an anonymous user ID, a connected cellular base station ID, a time stamp and an App ID. Each App ID is associated with one or more App type IDs. I sort the App usage records by ascending order of time. I use the first 80% of the records as training data and the last 20% as ground truth for testing.

System Setting: I set the default embedding dimension d and importance coefficient of user profile β as 20 and 0.5 respectively. I will also check how these two key parameters affect system performance in Section 3.5.2.

Performance Metric: I adopt *Accuracy@k* to evaluate prediction accuracy [53]. *Accuracy@k* is the statistical result of all test predictions, which is calculated with hit@k. The value of *hit@k* for a single prediction equals 1 if the ground truth App appears in the top k predictions, or 0 if otherwise. The overall *Accuracy@k* is calculated as the average over all test cases:

$$\text{Accuracy@k} = \frac{\#hit@k}{|R_{test}|}, \tag{3.7}$$

where $\#hit@k$ and $|R_{test}|$ represent the number of hits in the whole test set and the number of test cases.

Baselines: In order to illustrate the advantages of my system design, I compare my *CAP* with the following baselines.

- *Statistics (Sta):* This method counts the users' history of mobile App usage at the same time then selects the most frequently used one. This is an intuitive and direct method with time information for prediction.
- *Pure Statistics (P-Sta):* This method counts the users' history of mobile App usage and selects the most frequently used one. This is an intuitive and direct method for prediction, which does not use time and location information.
- *Graph based embedding (GE):* This method adopts the graph-based embedding in a recent work [53]. Besides the three graphs in my system, this section also embeds App-App sequential relationships. In addition, this method also adopts user profiles, but ones generated by APP usage history, current location, and current time, i.e. without App time decay. By comparing this method with my *CAP*, I can check the performance improvement against my embedding method and user profile.

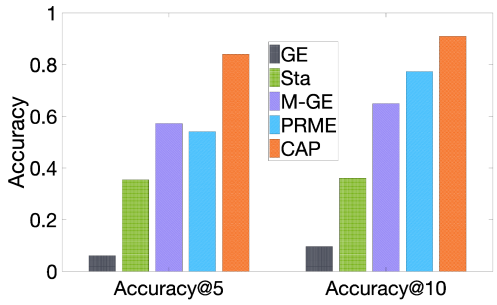


Figure 3.7: This figure shows Accuracy@5 and Accuracy@10 from our *CAP* and baselines. My *CAP* performs best, achieving 84% Accuracy@5 and 91% Accuracy@10. *M-GE* and *PRME* rank second and third. *Sta* and *GE* achieve low accuracy.

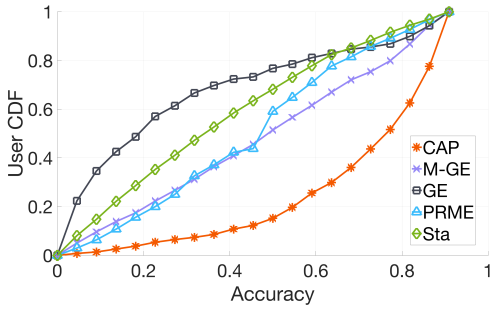


Figure 3.8: This figure shows the cumulative distribution function (CDF) of Accuracy@5 for all methods. Only 10% of users have Accuracy@5 less than 50% with our *CAP*. On the contrary, all baselines have at least 40% of users with Accuracy@5 less than 50%.

- *Modified graph based embedding (M-GE):* This method is a combinational scheme, which takes the same embedding method as my system and same user profile generation method as *GE*. By comparing this method with my *CAP*, I can check the performance improvement against my user profile.
- *Personalized Ranking Metric Embedding (PRME):* This method jointly models the sequential transition of App usage and user profile [54]. PRME utilizes one sequential transition space and one user profile space [54].

3.5.2 Result Analysis

In order to compare my *CAP* with baselines, I plot Accuracy@5 and Accuracy@10 in Figure 3.7. My *CAP* performs best, achieving 84% and 91% in terms of Accuracy@5 and Accuracy@10 respectively. *M-GE* and *PRME* rank second and third, both of which achieve around 30% lower accuracy than *CAP* in terms of Accuracy@5. *Sta* and *GE* achieve only 35% and 6% in terms of Accuracy@5 respectively. First, *Sta* does not get high accuracy since the simple statistical method cannot handle the cases when a user is going to open a new mobile App that does not show up in his/her training set. For example, in the testing set, user 0067461 opens App 59 at 7:43 pm at cellular tower 336271. The App 59 has never been used by this user and does not appear in the *Sta* prediction list. In contrast, the top 5 predictions from my

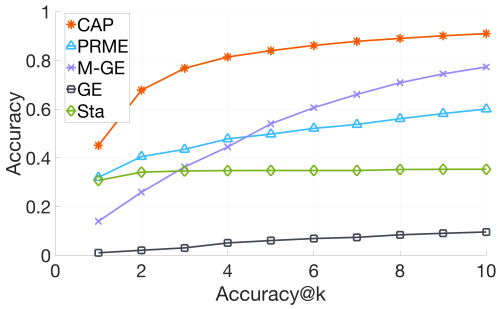


Figure 3.9: This figure shows Accuracy@k with different k values for all methods. Accuracy of *Sta* does not improve with larger k, while other methods illustrate increasing trend. My *CAP* achieves $\sim$ 80% in terms of Accuracy@3, which is higher than Accuracy@10 in terms of all other methods.

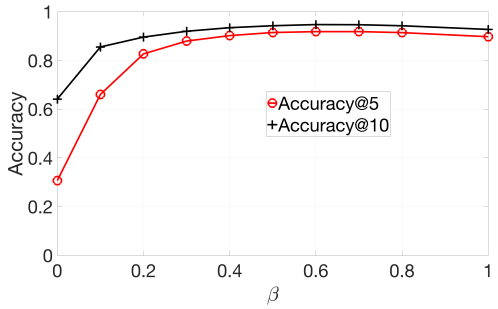


Figure 3.10: This figure shows the Accuracy@5 and Accuracy@10 with different user profile importance coefficient β values using my *CAP*. For both Accuracy@5 and Accuracy@10, the peak values show when $\beta = 0.5$. The accuracy at $\beta = 1$ is much higher than accuracy at $\beta = 0$.

CAP are App 179 59 241 125 144. This is because my *CAP* takes all users' history data and correlates App usage temporal, spatial and attribution characteristics through hetergeneous graph embedding. Second, *GE* performs worst since it includes App-App sequential relationship in embedding, which causes wrong connections. This illustrates that adding wrong interactions between influential factors leads to serious performance deterioration. In contrast, *M-GE* and *PRME* perform much better than *GE*, proving that what App a user is going to use is not decided by the previous used App, but instead by time and location. Third, the advantage of my *CAP* over *M-GE* validates the effectiveness of my user profile. The time-decay on both App usage and user's trajectory history help extract the dynamic user preference.

Figure 3.8 shows detailed statistics of Accuracy@5 for all methods. When using my *CAP* to predict, more than 50% users achieve Accuracy@5 higher than 80%. This proves that at least 50% of the users in my dataset are predictable, i.e. showing a specific behavior pattern. On the contrary, all baselines have at least 40% predictions with Accuracy@5 lower than 50%. This proves that my embedding scheme plus user profile have a strong ability to limit large prediction errors. In contrast, the inappropriate user profile from *M-GE* and inappropriate embedding scheme from *GE* cause large prediction errors.

To further analyze the accuracy of different metrics for different methods, I plot

Accuracy@k in Figure 3.9. The accuracy of *Sta* does not improve with larger k, while other methods illustrate increasing trend. This is because *Sta* lacks the ability to predict the mobile App usage at a location that a user has not visited or predict a future mobile App usage that a user has not used. On the contrary, all the other methods are able to achieve that based on the embedding scheme. This proves that embedding enables prediction in both these scenarios. In addition, my *CAP* achieves $\sim 80\%$ in terms of Accuracy@3, which is higher than Accuracy@10 for all other methods. In other words, with only three candidates, *CAP* is able to outperform other approaches with ten candidates. Only *M-GE* achieves similar accuracy in terms of Accuracy@8, Accuracy@9, and Accuracy@10. This illustrates that to achieve similar accuracy, my *CAP* requires a much smaller prediction list.

In order to illustrate how the user profile importance coefficient β in Eq.3.5 affects my system performance, I plot the Accuracy@5 and Accuracy@10 with different β values in Figure 3.10. For both Accuracy@5 and Accuracy@10, the peak values show when $\beta = 0.5$. This means that in the optimal solution of user profile, App usage and user's trajectory history play equally important roles. This validates my selection of user profile combination. In addition, accuracy at $\beta = 1$, when only the user's trajectory history is adopted, is much higher than accuracy at $\beta = 0$, when only the user's App usage history is adopted. This means that if only one factor can be included to represent a user's dynamic preference, his or her trajectory history is more important. This is because mobile App usage is more related to location than past usage.

Figure 3.11 shows how embedding dimension affects my system Accuracy@5 and Accuracy@10. First, obviously, high embedding dimension leads to high accuracy. Second, Accuracy@5 and Accuracy@10 saturates at 80 embedding dimensions, which achieves 94% and 97% accuracy respectively. This means that 80 dimensions are large enough to embed information. Third, the accuracy improvement shows an obvious difference before and after 20 embedding dimensions. From 5 embedding dimensions to 20 embedding dimensions, the prediction accuracy improves 56% (from 28% to 84%) and 35% (from 56% to 91%) in terms of Accuracy@5 and Accuracy@10 respectively. In contrast, from 20 embedding dimensions to 100 embedding dimensions, the prediction accuracy improves no large than 10% for both Accuracy@5 and Accuracy@10. Considering the tradeoff between accuracy and computing complexity, I adopt 20 embedding dimensions as the system default

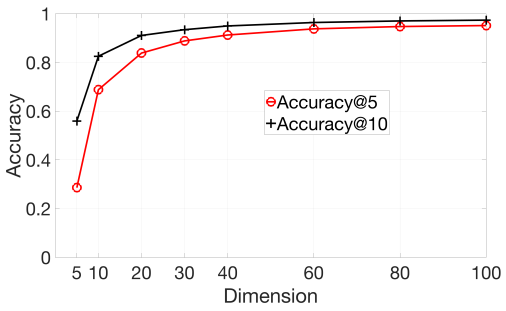


Figure 3.11: This figure shows Accuracy@5 and Accuracy@10 with different embedding dimensions using my *CAP*.

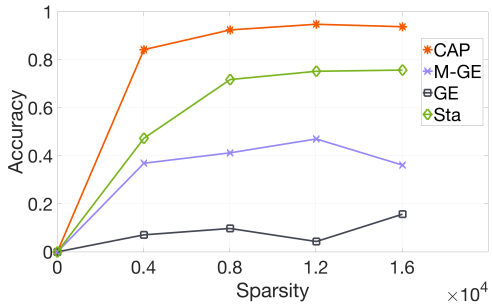


Figure 3.12: This figure shows Accuracy@5 with different user's data sparsity from different methods. More data help improve accuracy for all methods. My *CAP* saturates at 8000 with 92% Accuracy@5.

setting.

In order to show how a user's data sparsity affects accuracy, I plot Accuracy@5 from different methods in Figure 3.12. More data helps improve accuracy for all methods. This is because more history information gets better training, thus better predictions. My *CAP* saturates at 8000, which achieves 92% Accuracy@5. This illustrates that if all users have more than 8000 history records, my *CAP* can achieve up to 92% accuracy with five prediction candicates. On the contrary, *Sta*, *M-Ge* and *GE* saturate to get Accuracy@5 of $\sim 70\%$, $\sim 40\%$ and $\sim 10\%$.

In order to check the prediction accuracy of different users with the system, I plot Accuracy@5 with different user types in Figure 3.13. I classify users according to how many App types they use in the whole dataset. All user types achieve more than 70% in terms of Accuracy@5, which illustrates the robustness of the *CAP* on predicting for different user types. The robustness comes from my dynamic user profile, which includes the App usage and trajectory history.

Figure 3.14 shows Accuracy@5 and Accuracy@10 of different App type predictions with my *CAP*. All App types achieve more than 70% in terms of Accuracy@5 and 80% in terms of Accuracy@10. This illustrates the robustness of my *CAP* on predicting different App types. Taxi App ranks highest since users take taxis at regular locations and times, such as 8:00am from home, 3:00pm from school, 6:00pm from work etc. Office App ranks at the lowest accuracy since people in Shanghai, a big city in China, have high pressure on working and they could work at multiple time and locations.

In conclusion, this subsection evaluates the performance of my *CAP* and four

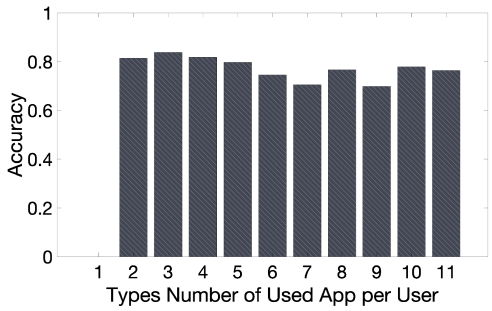


Figure 3.13: This figure shows Accuracy@5 with different user types. I classify users according to how many App types they use in the whole dataset. All user types achieve more than 70% in terms of Accuracy@5, which illustrates the robustness of my *CAP*.

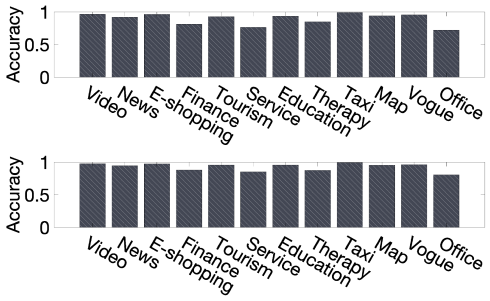


Figure 3.14: This figure shows Accuracy@5 and Accuracy@10 of different App type predictions with my *CAP*. All App types achieve more than 70% in terms of Accuracy@5 and 80% in terms of Accuracy@10. Taxi App ranks highest since users tend to take taxis at regular locations and times.

baselines to show the advantages of my system design. First, the extracted three interactions (App-time, App-location, App-App type) are proved to be valid on App usage prediction. The interactions between the influential factors need to be carefully designed. Adding App-App sequential interaction seriously deteriorate the performance. Second, the heterogeneous graph embedding is proved to successfully map the influential factors from different spaces into the same latent space and figure out the importance of these factors, which help predict App usage. Finally, the proposed user profile is shown to help improve the prediction accuracy by my method's accuracy improvement over *M-GE*.

3.6 Related Work

App Usage Behavior Modelling

Recent works have studied how users use mobile Apps by focusing on three aspects: user interactions, network traffic, and energy drain [24, 25, 26]. Church et al. summarized the challenges for mobile phone usage learning and analysis, as well as a series of studies and applications on mobile phone usage [27]. Falaki et al. discover immense diversity usage activities among users[24]. Another related work [55] reveals that users can be identified through the sets of Apps they use. Other studies cluster

mobile users according to their App usage records[56]. Moreover, users' mobility patterns can impact the way that the Apps are used [57]. Context such as location and time are shown to have impact on App usage [29][58]. A multi-faceted approach to predict App usage is developed in [32]. Most studies focus on small-scale datasets, posing a key challenges to understand and predict the App usage behavior over a large user population.

3.6.1 Recommendation Methods

Recommendation systems have been widely used and a wide range of approaches have been proposed. Context-aware recommendation is achieved by using additional information of location, time, and activity [33, 34, 35, 36, 37]. Zheng et al. and Karatzoglou et al. presented collaborative filtering based recommendation algorithm and use a large-scale user data pool to collaboratively filtering the like-minded users at different locations or activities [33, 35]. Zhu et al. focused on the problem of insufficient information from individual users by learning the common context-aware preference of many users, and the context sensors they targeted are spatio sensors such as GPS and accelerometer [34]. Kostakos et al. applied a Markov state transition model to predict next screen event [31]. Based on our study, the both spatio and temporal contextual information matters in the user behavior prediction. However, these prior works mostly limited the contextual information to location and activity. Zhao et al. [59] proposed a spatial-temporal latent ranking (STELLAR) method to explicitly model the interactions among user, POI, and time. Liu et al. considered both spatio and temporal contextual information and extended the RNN model to Spatial Temporal Recurrent Neural Networks (ST-RNN) with a time-specific transition matrices and a distance-specific transition matrices [37]. None of these works focus on App usage patterns. Compared to these prior works, my system CAP is able to project both context and attribute information into comparable spaces, hence achieving a better integration.

Other than contextual information, user profile information is also used to achieve personalized recommendation [39, 40, 41, 42, 43]. Rendle et al. presented their Factorizing Personalized Markov Chains (FPMC) model that subsumes both a common Markov chain and the normal matrix factorization model to profile the user. However, these personalized recommendation largely depends on the personal

profiling, which can be biased and may not capture the local trend of the App usage. Liu et al. used a mobile Customized Content Service (m-CCS) to filter blog articles to mobile users based on the trend of time-sensitive popularity of weblogs and the users' browsing logs to determine their interests [40]. Costa et al. monitors the users' interaction and made recommendation based on the users' friends, similar behavior users, and the similarity between Apps [41]. Similarly, Bohmer et al. leveraged the insights of users' engagement with particular applications to achieve recommendation [42]. Lin et al. presented PRemiSE, which takes into account potential influencers on virtual social networks extracted from implicit feedbacks for recommendation [43]. My work, compared to these prior works, allows better real-time modeling on personal choice of the App usage by combining the spatial and temporal contextual information in both group and individual level.

App similarity that is important for recommendation is calculate by graph [60] or kernel function [61, 62], which is utilized in Ranking [63] and popularity [64] based recommendation. When the user data is sparse, new challenges emerge. Problems of data sparsity [65] and cold-start [66] have been studied by using specific Apps' features of similarity. CAP handles the data sparsity by taking into account the data from many users in both spatio and temporal contextual information. Other recommendation has been done with a different focus from mine, which is privacy and security awareness [67, 68, 69]. These works demonstrate the possibility of secure aspects of recommendation systems like mine.

To summarize, none of the existing works focus on App usage prediction over a large population using both temporal and spatial information. These works have not explored the key influential factors & their interactions for App usage prediction, nor have they investigated the appropriate method to extract the importance of this information. In addition, the personalized factor of users has not been studied for App usage prediction. *CAP* achieves real-time personalized App usage prediction for multiple commercial applications as I discussed in Section 1. Context information (time & location), attribute information (App & App type) and the dynamic user preference are considered. I find that the relationships of App-location, App-time, and App-App type are essential to prediction and propose a heterogeneous graph embedding algorithm to map them into one common comparable latent space. I propose a user profile with his/her App usage & trajectory history affected by time decay factor, to achieve personalized prediction. I extract both the common

attribution of all users and individual user dynamic preference to ensure sufficient training data without losing personalization.

3.6.2 User Behavior Prediction

With the popularity of social networks, users leave a large volume of digital footprints online. By analyzing re-post behavior in social networks, Lu et al. [70] predicted the content dissemination trends. However, in traditional social networks, the user behavior such as posting blogs, sharing photos and uploading videos, does not necessarily reflect their daily activities. Location-based social networks (LBSNs), where users can share their real-time activities by checking in at POIs, provide a novel data source to study the collective behavior, and collective behavior analysis in LBSNs has gained increasing popularity in academia. For example, Cheng et al. [71] investigated 22 million checkins across 220,000 users and report a quantitative assessment of human mobility patterns by analyzing the spatial, temporal, social, and textual aspects associated with these footprints. Noulas et al. [72] conducted an empirical study of geographic user activity patterns based on check-in data in Foursquare. Cranshaw et al. [73] studied the dynamics of a city based on user collective behavior in LBSNs. Wang et al. [74] investigated the community detection and profiling problem using users' collective behaviors in LBSNs. In addition, the analysis of collective behavior in LBSNs can also enable various applications. For example, by analyzing users' check-in data in LBSNs, Yang et al.[75, 76] studied the personalized location based services such as POI recommendation and search. Sarwat et al. [77] introduced the Plutus framework that assists different POI (e.g., restaurants or shopping malls) owners in growing their business by recommending potential customers. Yang et al. [78] studied the large-scale collective behavior by introducing the NationTelescope platform to collect, analyze and visualize the user check-in behavior in LBSNs on a global scale. However, traditional LBSN cannot get access to mobile application data, so that predicting the App usage is a novel contribution of this section.

Mobility prediction is also widely studied. Markov model and its variations are common models to predict human mobility. Markov model [79, 80] consider the probability to capture the unobserved characteristics between location transition, i.e., Mathew et al. [81] cluster the locations from the trajectories and then train a

Hidden Markov Model for each user. Considering the mobility similarity between user group, Zhang et al. [82] propose GMove to share significant movement regularity among users.Moreover, pattern-based methods [83, 84, 85] also utilized to predict the mobility based on these popular patterns. All these mobility prediction techniques only deal with the two dimensional location and time information. However, in my context-aware App usage prediction, high-dimensional dataset are needed to be considered, which is a much more challenging problem.

3.7 Conclusion

This chapter presents *CAP*, a context-aware personalized App usage prediction system that takes both contextual information (location & time) and attribution (App & App type) information into consideration. I find that the relationships between App-location, App-time, and App-App type are essential to this prediction and I utilize spatiotemporal relationship (ST-SYS) of different influential factors from different human beings to design a heterogeneous graph embedding algorithm to map them into one common comparable latent space. I design a personalized user profile with his/her historical App usage and trajectory to describe individual dynamic preferences. I evaluate my system based on a large-scale real-world dataset, which includes more than 6 million mobile App usage records from 1788 individual users. The evaluation validates 1) the designated three direct and indirect interactions between influential factors of App usage, 2) adopting heterogeneous graph embedding to map these influential factors and 3) the proposed user profile. The results show that *CAP* achieves 35% higher accuracy than a related work method *PRME* in terms of Accuracy@5. At the same time, prediction with three candidates using my *CAP* achieves higher accuracy than prediction with ten candidates using all the other baselines. All these results validate the effectiveness of applying the spatiotemporal relationship (ST-SYS) on the large scale CPS for system status estimation.

Chapter 4

Estimating the Physical Status for Large Scale Mobile CPS

This chapter introduces how to utilize spatiotemporal relationships to address the challenge of sparse coverage and uneven distribution in physical field estimation (air pollution estimation), as described in Chapter 1. Besides the system status estimation in Chapter 3, estimating the physical field is another essential task of field estimation, which helps the system to understand a physical process based on the sensing data. However, purely applying traditional data-driven methods, especially machine learning methods, to estimate leads to bad and unstable performance. This is because the performance of purely data-driven methods is highly biased by the data collection and suffers from poor and unstable performance when the data collection is insufficient and time-variant. Even with large scale mobile devices, the system is still not able to guarantee that sufficient information is collected. Take the example of vehicle based CPS, where most vehicles gather in the central areas where most human activities happen, thus collecting redundant information. Thus the outskirts, large parts of the city, are not well covered by the vehicle fleet. In addition, the high mobility of devices means the spatial coverage changes over time, making physical field estimation challenging for traditional data-driven methods.

To address the challenges, the evolution of the physical field over space and time (ST-PHY in Chapter 2), determined by the laws of physics and described as a physics guided model, is adopted to help estimate the physical field. The physics guided model indicates the relationship of values at adjacent areas and time

periods, introducing (temporal and spatial) interactions of the collected data and offering extra information for physical field estimation. In addition, the ST-PHY also prevents the estimation from being biased by the data collection. Moreover, to ensure stable and accurate physical field estimation, features such as the amount of data collection and the estimation quality at different areas and times (ST-SYS) are used to adaptively switch between the physics guided model and the data driven model. Another way to address the challenges are dispatching mobile devices to collect data with dense coverage and even distribution, which will be discussed in Chapter 5 (the controllable system) and Chapter 6 (the semi-controllable system).

The system introduces the temporal and spatial interaction of air dispersion with a physics guided model (ST-PHY), which offers extra information when the sensing coverage is very sparse. The system adaptively fuses the results from the physics guided model and the data driven model based on features such as the amount of data collection and the estimation quality at different areas and times (ST-SYS). This guarantees the stability of the system even as the sensing coverage changes over time and location.

I first motivate the problem in Section 4.1. Then Section 4.2 introduces the technical background of the section. After that, I describe the system design in Section 4.3 and algorithm design in Section 4.4. The evaluation is discussed in Section 4.5. Finally, I introduce the related work in Section 4.6 and conclude the section in Section 4.7.

4.1 Problem Overview

Air pollution has become a major public health crisis worldwide [12, 13]. According to the World Health Organization (WHO), every year more than 3 million deaths are linked to exposure to outdoor air pollution and 92% of the world's population lives in places where air pollution levels exceed WHO limits [14].

It is essential to provide fine-grained air pollution information in both time and location to both residents and city managers (e.g. such as block level at every hour) [86]. This information helps city managers to understand pollution and sources for further management, planning and development [19]. In addition, an air pollution map at block scale helps residents understand and reduce their exposure risks (e.g.

choosing the commute routes or housing with low pollution) [15, 16].

Prior research has investigated using mobile sensing platforms to improve coverage and resolution over static approaches [87, 88]. Sensors are installed on vehicle fleets, especially taxis to collect air pollution data with improved cost, and mobility. To obtain fine-grained air pollution maps with mobile collected data, there has been two main approaches. The first is a physics guided approach, which adopt physical principle to describe how air evolves over time and space [89, 90, 91]. Given the equation of physical principle, these methods can calculate very high resolution pollution maps. However, these approaches generally have low accuracy due to 1) difficulty to capture all possible factors that influence air pollution, such as geographical information like buildings, pollution sources etc., and 2) inaccurate empirical parameter assumption or estimation for dynamic urban environment. The second approach is data driven, which uses sensed air pollution data and other information (such as traffic, weather etc.) to derive air pollution maps using one or more data-driven models [87, 88, 92, 93, 94]. The accuracy of these methods is high when sufficient data is collected, but deteriorates when sensing coverage is sparse. This limits the time and location where these approaches can be applied.

Major challenges: sparse and time-variant sensing coverage. *(i.) Sparse Sensing Coverage:* Since most mobile sensing nodes like vehicles or taxis gather around busy areas, it leaves large parts of the city not sensed at any given time. Consequently, the data density for many parts of the city is insufficient. *(ii.) Time-variant Sensing Coverage:* The sensed and non-sensed areas change significantly over time due to uncontrollable movements of these mobile nodes. As a result, the time-variance of sensing coverage leads to the unstable performance at different times.

This chapter presents a physics guided and adaptive (PGA) approach to derive an air pollution map with high spatial (block level) and temporal (hourly) resolution. I utilize features such as the amount of data collection and the estimation quality at different areas and times (ST-SYS) to design an adaptive system structure, which combines the advantages of high resolution from a physics guided model and high accuracy from a data driven model using a particle filter. To address the challenge of sparse sensing coverage, the system infers information at the uncovered areas based on the temporal and spatial relations of air dispersion with a physics guided model. To address the challenge of time-variant sensing coverage, PGA adaptively selects

combination information from either the physics guided model or data driven model at different time and locations. The intuition here is the system weighs more on the physics guided model during the period when data coverage is sparse and weighs more on the data driven model during the period when data coverage is dense.

The main contributions of this section are:

- Reformulating a physics guided model for air dispersion state evolution estimation and combining it with a data driven model.
- Proposing an adaptive scheme to correct estimate from a physics guided model with estimation generated from a data driven model.
- Deploying and evaluating the system with Particle Filter structure on a large-scale vehicular sensing platform for large-scale evaluation.

To evaluate the system, I deploy 29 taxis with low cost ($\sim$ 1000 USD for each device) air pollution sensing devices in the city of Shenzhen for 14 days, which collects around 26.3 million data samples. The evaluation results show with resolution of 500 m by 500 m by 1 hour, my system achieves up to 5.0$\times$ and 2.1$\times$ reduction on average error, as well as up to 8.0$\times$ and 3.7$\times$ improvement on stability, compared to artificial neural network and another state-of-the-art combination approach [87]. To achieve similar performance, my approach requires at least 5$\times$ less sensor deployment than other approaches.

4.2 Physics Guided Background

This section provides background for two main techniques in this section: the physics guided model and the system structure design. I first discuss advantages of adopting a physics guided model to deal with the sparse sensing coverage problem. Then, I introduce the background of a structure, based on what I design my system to address time-variant sensing coverage problem.

4.2.1 Physics Guided Model

The physics guided model I build upon in this section describes how a physical field, such as air particles, temperature etc., evolves over time and space. In particular, the NavierâĂŞStokes equations describe the motion of viscous fluid substances [95, 96].

Convectional diffusion equation is widely used to describe the transmission of particles energy and other physical quantities including air dispersion [97, 98].

The convectionâĂŞdiffusion equation is a good fit for deriving a fine-grained air pollution map. This is because it reveals the dynamics of the air evolution and describes the relations of quantities at different times and locations [98]. These relations help to infer air pollution information at the uncovered areas to derive fine-grained air pollution map when the sensing coverage is too sparse.

The form of the convectionâĂŞdiffusion equation is shown in Equation 4.1. In the equation, $C[\text{kg/m}^3] = C(x, y, z, t)$ is the gas (air pollutant) concentration at location (x, y, z) and time t. $S[\text{kg/m}^3\text{s}]$ represents the environmental factor parameters. There are two parameters in the equation: the wind velocity vector $\boldsymbol{v} = (v_x, v_y)$ [m/s] and the diffusion coefficient $\mathbf{K} = \text{diag}(K_x, K_y, K_z)$ [m^2/s], which is a diagonal matrix with entries representing the turbulent eddy diffusivities [97].

In equation 4.1, the temporal changing rate of air pollution $\frac{\partial C}{\partial t}$ consists of three contributions. The first term $\nabla \cdot (K \nabla C)$ describes the pollutant flux caused by diffusion. The second term $\nabla \cdot (\boldsymbol{v} C)$ describes the pollutant flux caused by the wind. The third contribution S describes the local creation or destruction of the air pollution. Due to the continuity or conservation of air pollutants, I obtain the final form as shown in Equation 4.1:

$$\frac{\partial C}{\partial t} + \nabla \cdot (\boldsymbol{v} C) = \nabla \cdot (K \nabla C) + S. \tag{4.1}$$

However, directly applying the model alone to obtain air pollution map is infeasible. First, the equation is not computationally practical due to the differential form even after discretization. The equation does not explicitly derive the current air pollution $C(t)$ given all $C(\tau)$, where $\tau < t$. Second, the initial air pollution state is only partially known due to the sparse sensing coverage. This will lead to aggregated bias as time progresses. Finally, the environmental factors in the equation change over time, which cannot be captured accurately in real time. The real time estimation introduces errors, which propagate over time and space.

To integrate the physics guided model in my system, I first reformulate Equation 4.1 for air pollution state evolution estimation (discussed in Section 4.4.1). Then I introduce a data-driven model to correct the estimation errors from the physics guided model. These two models cooperate using a Particle Filter structure, which

works with an adaptive scheme to deal with both sparse and time-variant sensing challenges. I further describe the particle filter structure next, in Section 4.2.2 and how I design adaptive scheme in Section 4.4.

4.2.2 Structure Design Background

PGA is built using a Particle Filter structure to combine the physics guided model and the data driven model. Here I provide a brief background of the particle filter approach that is relevant to my system.

Particle Filter Design: A Particle Filter is a sequential Monte Carlo method for on-line state tracking, which works within a Bayesian framework and under Markov assumption [99, 100]. It uses a finite number of elements to represent a non-parametric probability density function (PDF) of the estimated state [101]. A Particle Filter is a recursive filtering consisting of two stages: 1) **state evolution estimation** which predicting next state PDF from the current estimate and 2) **estimate correction** which corrects the prediction using new measurements.

The Particle Filter structure is particularly well suited for deriving an air pollution map. First, the air pollution evolution fits the hidden Markov assumption since given the pollution at previous time slot, the current pollution level only depends on the pollution level in the previous time slot. Second, I can not use a simpler distribution (e.g. Gaussian distribution) based on the experimentation results. Finally, similar to the particle filter, I utilize a state evolution estimation (physical model) and estimation correction (adaptively correct physical model with data model). The details can be found in Section 4.4.

To design a Particle Filter, two probabilistic models should be available for state evolution estimation and estimate correction [99]. If the state to be estimated at time t is indicated by the vector y_t, the model for **state evolution estimation** can be denoted as

$$p(y_t|y_{t-1}, u_t), \tag{4.2}$$

where u_t represents the cause of evolution from $t-1$ to t. To be specific in deriving an air pollution map, u_t includes environmental factors at time t that causes the air dispersion.

If the new measurement at time t is denoted by z_t, the model for estimate

correction is represented by

$$p(z_t|y_t). \tag{4.3}$$

Since Particle Filter holds a Markov assumption, y_t is conditionally independent from y_k when $k < t-1$ once y_{t-1} is known. Similarly, the measurement z_t is conditionally independent from any other variables if y_t is known. I do not have to know specific distributions in Eq. 4.2 and Eq. 4.3.

The algorithm initializes with a set of N particles $\mathcal{Y}$, representing the posterior of y_{t-1}. Given the latest u_t and measurement z_t, it generates a new set of N particles, which represents an update of the posterior estimate for y at time t. The ith particle in $\mathcal{Y}_t$, $y_t^{[i]}$, provides the ith possible hypothesis about the state at time t [101].

Inadequacy of a Simple Particle Filters: A naive way to implement the *PGA* system would be to directly use the physics guided model for **state evolution estimation**, while the data driven model generates measurements from collected data for **estimate correction**. The gas (air pollutant) concentration to be estimated is y_t, while the generated measurements are z_t. u_t is the new air pollution sources.

Directly applying the Particle Filter structure to combine the two models does not necessarily guarantee performance improvement. This is because time-variant sensing coverage causes the quality of data-driven model outputs to vary significantly over time and location. When the collected data coverage is highly sparse, the measurements generated by the data driven model is unreliable. As a result, correcting the estimate with these measurements can deteriorate the overall performance. Therefore, I present an adaptive model fusion structure to estimate the quality of the results before the correction. The details of the adaptive scheme are described in Section 4.4.

4.3 System Design

This section introduces how I design system architecture to address the key challenges of *the sparse sensing coverage and time-variant sensing coverage*. Figure 4.1 shows the high-level architecture of the vehicle based CPS for physical field estimation (air pollution estimation). Air pollution information is collected through sensors on the vehicles from the *information sensing layer*. The collected information is stored in the *database* and processed in the *information estimation layer*. The *information*

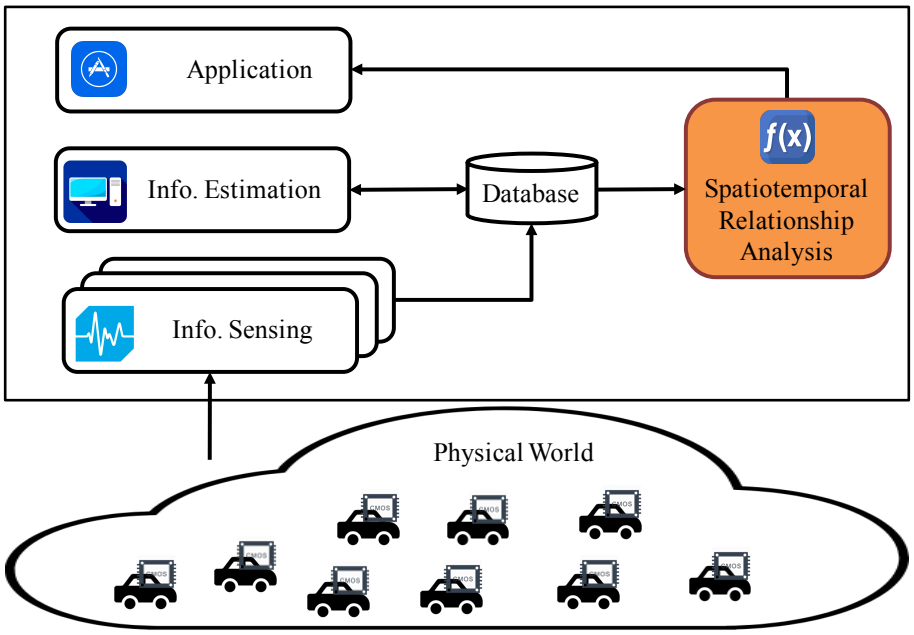


Figure 4.1: The figure shows the system architecture of spatiotemporal relationship aided large scale vehicle based CPS for physical field esimation (city-scale fine-grained air pollution estimation).

estimation layer adopts a data-driven model to infer the air pollution and stores the results back to the *database*. The *spatiotemporal relationship analysis layer* adopts both the raw data from the *information sensing layer* and the processed results from the *information estimation layer* to improve the estimation results and sents the results to the *application layer*. The detailed design of the *spatiotemporal relationship analysis layer* is shown in Figure 4.2.

Pre-Processing Module: According to resolution requirement, the pre-processing module discretizes the temporal and spatial domain into n_{lon} by n_{lat} congruent *sub-areas* (x_i, y_j) and *time slices* t_k. x_i, y_j, t_k correspond to longitude, latitude and time index respectively. The size of each sub-areas is $\Delta x \times \Delta y$ and $\Delta x = \Delta y = l$. The *sub-areas* and *time slices* combine to form the smallest of the system *unit* (x_i, y_j, t_k), which defines my pollution map resolution. When the cloud receives data from mobile nodes, the pre-processing module puts them into different units. If multiple data exists in the same unit, the system adopts the average value to represent that unit for further processing. In order to reduce the computational complexity, rather than using a centralized Particle Filter jointly estimating the air pollution of all the *sub-areas*, I associate a Particle Filter to each *sub-area*. It is noticed that the

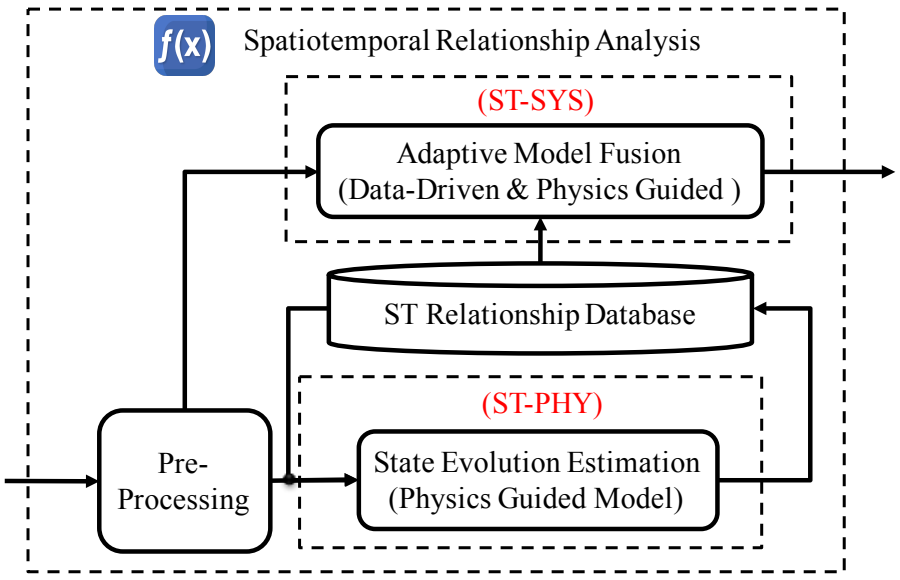


Figure 4.2: This figure shows the architecture of the *spatiotemporal relationship analysis layer*, which utilizes 1) spatiotemporal relationship of air pollution (ST-PHY) and 2) the amount data collection and estimation quality at different areas and times (ST-SYS), to help the large scale mobile CPS (vehicle based CPS) for physical field estimation (city-scale fine-grained air pollution estimation).

particle filters at different *sub-areas* are not independently calculated. Instead they interact with each other through the physics guided state evolution estimation, since the temporal evolution of air pollution involves air exchange over spatial domain, as shown in Eq. 4.1.

State Evolution Estimation: To address the sparse sensing coverage problem, the system takes advantage of the spatial and temporal relations of air dispersion shown in equation 4.1. The physics guided model allows the system to 1) obtain high temporal-spatial resolution and 2) maintain air pollution field information when the sensing coverage is very sparse. Since it is difficult to predict the next state from the current estimate with this differential equation, I first reformulate Equation 4.1 to a predictable form $(C(t) = f(C(t-1)))$. Then I integrate the reformulated equation in the Particle Filter structure for state evolution estimation. This module takes the estimation output at *time slice* t_{k-1} as input and applies the reformulated form of the physics guided model to estimate the air pollution at at *time slice* t_k. In addition, before the state evolution estimation is conducted at each *time slice*, the environmental factor parameters S are estimated from the collected

data. The derived physics guided model and estimation from the model is stored in the *spatiotemporal relationship database*. I further discuss the details of this module in Section 4.4.1.

Adaptive Model Fusion: Due to the dynamics of the sensing coverage, neither the physics guided model nor the data-driven model from the *information estimation layer* can ensure the advantage over all the *sub-areas* and *time slices*. An adaptive model fusion scheme is designed to fuse the results from two models to guarantee that the system can choose the one with better accuracy. First, due to high required resolution and limited number of vehicles, most *sub-areas* do not have real data measurements. Therefore, I must generate data for estimate correction using the data-driven model in the *information estimation layer*. Second, whether the generated measurements have similar value as ground truth depends on whether there are real collected data nearby (close in proximity and time). When the generated measurements are near the real collected data, the generated measurements would be similar to ground truth values, and vice versa. Correcting the estimation with these generated measurements likely improve the accuracy. Third, since the vehicles are mobile, these sparse locations change and thus *PGA* must make this adaptation on-the-fly. Therefore, at different *sub-areas* and *time slices*, the system adaptively decides to conduct or skip the estimate correction according to whether or not the correction with generated measurements improves the performance. After the adaptive model fusion, the system gives each *sub-area* a distribution representing the possible values and their weights. Higher weights mean higher possiblilities. The distribution not only represents the possible real air pollution value hypothesis but also maintains an assessment of the likely accuracy of the estimate [102]. The final pollution value of each *sub-area* is the weighted sum of the distribution. In addition, the historical estimation results will be conveyed into the physics guided state evolution estimation module for the next state estimation. I further discuss the adaptive scheme in Section 4.4.2.

4.4 Algorithm Design

This subsection provides the algorithm description in the key modules in Figure 4.2: state evolution estimation (physics guided model) and adaptive model fusion (data-

driven & physics guided). These two modules are designed to address the two main challenges in this section: the sparse sensing coverage and time-variant sensing coverage problem respectively.

4.4.1 Reformulating the Physics Guided Model for State Evolution Estimation

In order to address the challenge of sparse sensing coverage, I introduce extra information from the temporal and spatial relations of air dispersion principle [97, 98]. To achieve this, I reformulate the physics guided model (shown in Equation 4.1), to provide a **state evolution estimation**. This state evolution estimation allows the system to infer physical values when (and where) the physical sensors is absent. This approach bases the estimation on physical laws that limits potential errors, when compared to a data-based approach, especially when the data samples are not a fair representation of the underlying distribution [103]. In addition, since the original physics guided model Equation 4.1 is expressed as a differential form, it must be transferred into a predictable form ($C(t) = f\left(C(t-1)\right)$) for a state evolution model.

Assumptions: To make the model computationally efficient, I make the following simplifying assumptions while still keeping temporal and spatial correlation: First, I focus on the air pollution near the ground, thereby bypassing the altitude variations in the air pollution field. Thus, I assume that $C(x, y, z) \equiv C(x, y)$. This assumption is valid for near ground concentrations when *sub-area* length ($500m$) is much greater than altitude changes ($<= 10m$) [90]. Second, since the time resolution in my application is one hour, and I evaluate the wind on a macro-scale ($500m$), I treat the factor of wind as a whole within an hour, thus the wind vector is estimated once every hour [90]. Third, different boundary conditions should be assumed in different areas, which are only decided by the situation around the marginal area. Since I apply my system in the center area of Shenzhen, China, whose boundaries are flat land and ocean and no large boundries exist, I assume an open boundary condition which fits the area I sense in my deployment and evaluation. For other areas, the boundary condition needs to be modified for implementation. Finally, the dispersion rate K in each direction is the same and independent of location, i.e. $K_x = K_y = K_c$, where K_c is a constant. From my experiments, I observe that this assumption have little impact on accuracy. It is easy to modify the corresponding

model if environmental study suggests the need for more complex forms of K_x and K_y. The changes will not affect the other parts of the algorithm [90].

It is noticed that the physics guided model only captures the major factors that air pollution evolves over time and space. More detailed factors, such as buildings in the area, small pollution sources like human beings, are impossible be included. Therefore, a data-driven model is needed to capture more detailed information.

Equation Derivation: With above assumptions, I can rewrite Equation 4.1 as

$$\frac{\partial C}{\partial t} + v_x \frac{\partial C}{\partial x} + v_y \frac{\partial C}{\partial y} = K(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}) + S, \tag{4.4}$$

where C is the gas (air pollutant) concentration. v_x and v_y denote wind velocity on X and Y directions. K and S are diffusion coefficients and local creation/destruction of air pollution. As mentioned in Section 4.2.1, the temporal changing rate of air pollution $\frac{\partial C}{\partial t}$ consists of three contributions. The first term $v_x \frac{\partial C}{\partial x} + v_y \frac{\partial C}{\partial y}$ describes the pollutant flux caused by the wind. The second term $K(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2})$ describes the pollutant flux caused by diffusion. The third contribution S describes the local creation or destruction of the air pollution.

While this model describes the physical properties of air dispersion with high fidelity, it cannot be employed for the state evolution estimation. This is because the model is illustrated in a continuous form while the data collected by the mobile sensing system comes from discrete samples in space and time.

In order to achieve computational practicality while maintaining the fidelity of the air diffusion process, I discretize Equation 4.4 on both temporal and spatial domain. As described in section 4.3, the pre-processing module splits the overall area into n_lat by n_lon congruent *sub-areas* and discretizes the time into equal *time slices*. The size of each sub-areas is $\Delta x \times \Delta y$ and $\Delta x = \Delta y = l$. Equation 4.4 can be rewritten as

$$\frac{\partial C}{\partial t} = K\left(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}\right) - v_x \frac{\partial C}{\partial x} - v_y \frac{\partial C}{\partial y} - C\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right) + S. \tag{4.5}$$

For ease of the analysis and the application, the above model is discretized in the spatial domain. The spatial discretization form of C is denoted as D and shown as follows,

$$\begin{pmatrix} D(1,1) & D(1,2) & ... & D(1,n_{lat}) \\ D(2,1) & D(2,2) & ... & D(2,n_{lat}) \\ ... & ... & ... & ... \\ D(n_{lon},1) & D(n_{lon},2) & ... & D(n_{lon},n_{lat}) \end{pmatrix}$$

I use the finite difference to replace the partial differential with respect to x and y in the above equation to yield to the following equation

$$\begin{aligned} &\tfrac{\partial D(i,j)}{\partial t} = K\left\{\tfrac{1}{\Delta x^2}[D(i+1,j)+D(i-1,j)-2D(i,j)] + \tfrac{1}{\Delta y^2}[D(i,j+1)+D(i,j-1)-2D(i,j)]\right\} \\ &-\tfrac{v_x(i,j)}{\Delta x}[D(i+1,j)-D(i,j)] - \tfrac{v_y(i,j)}{\Delta y}[D(i,j+1)-D(i,j)]- \\ &D(i,j)\left\{\tfrac{v_x(i+1,j)-v_x(i,j)}{\Delta x} + \tfrac{v_y(i,j+1)-v_y(i,j)}{\Delta y}\right\} + S(i,j). \end{aligned} \tag{4.6}$$

Then, I define a new $n_{lon} \cdot n_{lat}$ by $n_{lon} \cdot n_{lat}$ matrix A', and I use $\mathbf{A}'((i_1,j_1),(i_2,j_2))$ to denote $\mathbf{A}'((i_1-1)n_{lat}+j_1,(i_2-1)n_{lat}+j_2), \forall i_1,i_2 = 1,2,\cdots,n_{lon},\ j_1,j_2 = 1,2,\cdots,n_{lat}$. A' is a sparse matrix with non-zero values at

$$\mathbf{A}'((i,j),(i,j)) = -\frac{4K}{l^2} - \frac{1}{l}\left[v_x(i+1,j) - 2v_x(i,j) + v_y(i,j+1) - 2v_y(i,j)\right]$$

$$\mathbf{A}'((i,j),(i,j-1)) = \mathbf{A}'((i,j),(i-1,j)) = \frac{K}{l^2}$$

$$\mathbf{A}'((i,j),(i,j+1)) = \frac{K}{l^2} - \frac{v_y(i,j)}{l}$$

$$\mathbf{A}'((i,j),(i+1,j)) = \frac{K}{l^2} - \frac{v_x(i,j)}{l}.$$

To get the corresponding discrete form expression, matrix D is reshaped to form a $n_{lon} \cdot n_{lat}$ vector C', where

$$C'((i-1)n_{lat}+j)D(i,j) \quad \forall i = 1,2,\cdots,n_{lon}, j = 1,2,\cdots,n_{lat}.$$

Similarly, matrix S is reshaped to form a $n_{lon}\cdot \cdot n_{lat}$ vector S'. Then I could get:

$$\frac{dC'(t)}{dt} = A'C'(t) + S', \tag{4.7}$$

where $C'(t)$ is a $n\_lat \cdot n\_lon$ vector with its elements giving the air pollution value at different *sub-areas*, and S' is a $n\_lat \cdot n\_lon$ vector discretized from S

in equation 4.1. A' is a $n\_lat \cdot n\_lon$ by $n\_lat \cdot n\_lon$ matrix representing the dispersion dynamics, i.e., how the air pollution at different *sub-areas* influence each other.

After the spatial discretization, I further discretize equation 4.7 on temporal domain. Note that Equation 4.7 is an ordinary differential equation (ODE) [104]. I solve the equation to obtain the relationship between $C'(t + \Delta t)$ and $C'(t)$:

$$\frac{d}{dt}[\exp(-tA')C'(t)] = \exp(-tA')\left[\frac{dC'(t)}{dt} - \mathbf{A}'C'(t)\right] = S'\exp(-tA')$$

$$\exp(-(t+\Delta t)A')C'(t+\Delta t) - \exp(-tA')C'(t) = \int_t^{t+\Delta t} S'\exp(-\tau A')d\tau$$

$$\exp(-\Delta tA')C'(t+\Delta t) = C'(t) + \exp(tA')\int_t^{t+\Delta t} S'\exp(-\tau A')d\tau$$

. Rearrange the above terms would give:

$$C'(t+\Delta t) = \exp(\Delta tA')C' + [\exp(\Delta tA') - I]A'^{-1}S'. \tag{4.8}$$

Denote $A := \exp(\Delta tA'), B := [\exp(\Delta tA') - I]A'^{-1}$, I get the final discrete space and time air dispersion model characterized by the following equation:

$$\mathbf{C}[k+1] = A\mathbf{C}[k] + BU[k], \tag{4.9}$$

where k is the *time slice* index, and $\mathbf{C}[k]$ (n_lon by n_lat matrix) is the discretized form of $C'(t)$. $U[k]$ is the discretized form of S'. The coefficient matrix A describes the temporal and spatial correlation of the air pollution concentration while coefficient matrix B reveals the effect of environmental factor $U[k]$.

I apply particle filter on the elementwise representation of the matrix form formula 4.9 for state evolution estimation:

$$\mathbf{C}[k+1]^l_{i,j} = (A\mathbf{C}[k])^l_{i,j} + (BU[k])^l_{i,j} + \phi, \tag{4.10}$$

where $\mathbf{C}[k+1]^l_{i,j}$ is the lth particle at *sub-area* (i,j) and $(k+1)th$ *time slice*. ϕ is the noise of state evolution estimation, which is drawn from a distribution derived from real measurement. I implemented my algorithm based on this.

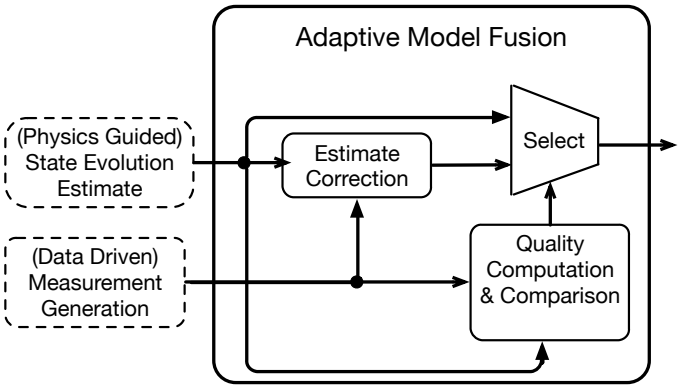


Figure 4.3: This figure shows detailed architecture of the adaptive model fusion scheme. The system adaptively chooses to correct physics guided state evolution estimation with generated measurement or not. The decision is based on comparing qualities of physics guided state evolution estimation with generated measurement.

It is noticed that given a *time slice* k, the key parameters to be estimated are the time-variant environmental factor, which is the $U[k]$ in equation 4.10. In my system, I use the air pollution measurement at and before *time slice* k to estimate the $U[k]$ through convex optimization [105, 106]. After $U[k]$ is estimated, Eq. 4.10 is totally decided and can be used to estimate the air pollution values at all *sub-areas*.

4.4.2 Adaptive Model Fusion Based on Quality Comparison:

In order to account for the time-variant sensing coverage, I design an adaptive model fusion scheme. This subsubsection first introduces the structure of adaptive model fusion scheme. Then I explain why and how I generate the measurements when no sensing data is present for estimate correction. Finally, I discuss how the adaptive decision is made.

Adaptive Model Fusion Architecture: In order to address the time-variant sensing coverage problem, I design an adaptive model fusion scheme as shown in Figure 4.3. The results of the state evolution estimation, which are stored in the *spatiotemporal relationship database*, are fed into the *estimate correction* unit for performance improvement. However, at any given time, with many more *sub-areas* than mobile sensing nodes, many *sub-areas* are not sensed. I generate measurements at these "empty" *sub-areas* with *(data driven) measurement generation* unit. This *(data driven) measurement generation* is performed in the *information estimation*

layer and its results is stored in the *database*. However, estimate correction with these generated measurements does not guarantee improved estimate performances. This is because when the *sub-area* is physically located far from *sub-areas* with real collected data, the generated measurement at the "empty" *sub-area* can have large difference from true value. As a result, conducting estimate correction at these *sub-areas* and *time slices* will likely to cause performance deterioration.

Therefore, at each *sub-area* and *time slice*, the system first computes and compares the quality of outputs from *physics guided state evolution module* and *data driven measurement generation* unit. This is done at *quality computation & comparison* unit. After that, the system decides to choose the outputs either before or after estimate correction. How the system makes the decision will be explained in Section 4.4.2.

Data Driven Measurement Generation: Since most *sub-areas* are not sensed at a given *time slice*, as unfilled squares, measurements for estimate correction at these *sub-areas* are unavailable. As a result, the errors from the physics guided state evolution estimation module could broadcast over time and space due to the model approximation error and environmental factor estimate error.

Therefore, in order to make estimate correction possible at all *sub-areas*, I use a data-driven method to generate the measurement at these empty *sub-areas*. My system adopts an Artificial Neural Network(ANN) for measurement generation due to its high capability to learn complex non-linear relationship [94, 107]. The details of implementation of the ANN could be found in [87, 94].

After the measurements are generated, it is possible to correct the estimate at each *sub-area* to get the new weight assignment [100] by

$$p(z_t|y_t) = f_{N,{\sigma_s}^2}(z_t - {y_t}^{[i]}), \tag{4.11}$$

where z_t and y_t represent generated measurement and state evolution estimation output respectively. $f_{N,{\sigma_s}^2}$ is the density probability of data measurement noise, which is obtained from experimentation. The Equation 4.11 needs to be discretized on both temporal and spatial domain. After all weights have been computed, resampling step [100] can take place to reassign the importance of particles according to their similarities to new measurements.

Quality Computation & Comparison: A key problem from sparse and time-variant sensing coverage is that consistently correcting estimates at all *sub-areas*

and *time slices* do not necessarily improve performance. This is because at some *sub-areas* and *time slices*, the generated measurements have very large difference from ground truth values. Using these low quality generated measurements to correct the estimates only lead to performance deterioration.

As a result, my adaptive correction scheme chooses to correct the estimation or not based on quality comparison to achieve the least error at each *sub-area* and *time slice*. With adaptive correction, the system has two choices: 1) conducting the correction with the generated data from the data driven model if the quality of the generated data is higher than the quality of estimation or 2) skipping the correction after estimation if the quality of the generated data is lower than the quality of estimation.

I use entropy of the estimation distribution to estimate quality of state evolution estimation [101, 108]. This is because entropy represents the number of bits required for optimal encoding, which can be used as an indication of the uncertainty of the estimation. I define **entropy of physics guided state evolution estimate distribution** $E[k]_{i,j}$ at *sub-area* (i, j) and *time slice* k as

$$E[k]_{i,j} = -\sum_{l=1}^{N} \left(p[k]^{l}_{i,j} \log_2 p[k]^{l}_{i,j} \right), \tag{4.12}$$

where $[k]^{l}_{i,j}$ is probability of lth particle for state evolution estimation distribution at *sub-area* (i, j) and *time slice* k, and N is the number of particles. The lower entropy $E[k]_{i,j}$ indicates higher quality of state evolution estimation, and vice versa.

The quality of generated measurement can be inferred from the distance between generated measurement and real sensed data. Usually, measurements generated by sensed data from farther *sub-areas* cause higher errors. Therefore, I define **sum of M nearest neighbours' distances** at *sub-area* (i, j) and *time slice* k as

$$D[k]_{i,j} = \left(\sum_{m=1}^{M} \sqrt{\left(i - i_{m[k]}\right)^2 + \left(j - j_{m[k]}\right)^2)} \right) \Bigg/ M, \tag{4.13}$$

where $(i_{m[k]}, j_{m[k]})$ represent the nearest M sensed data for *sub-area* (i, j) at *time slice* k. Smaller $D[k]_{i,j}$ means smaller errors from measurement generation. If only n

$(n < M)$ nearest neighbours exist, $D[k]_{i,j}$ is calculated as:

$$\left(\sum_{m=1}^{n}\sqrt{(i-i_{m[k]})^2+(j-j_{m[k]})^2}+(M-n)*D_{\max}\right)\Big/M, \tag{4.14}$$

where $D_{\max}=\sqrt{N^2{}_{lon}+N^2{}_{lat}}$, which is the diagonal line length of the area. N_{lon} and N_{lat} are the discretized length of the map on longitude and latitude, as described in Section 4.3. Here I select $M = 8$, since one *sub-area* has at most 8 adjacent neighbors.

After calculating these two quality metrics at each *sub-area* (i, j) and *time slice* k, the system normalizes them with α to keep them at same order of magnitude for quality comparison. Based on the comparison result, the system chooses to conduct estimate correction if $E[k]_{i,j}$ - $\alpha * D[k]_{i,j} < 0$ or skip the estimate correction if $E[k]_{i,j}$ - $\alpha * D[k]_{i,j} > 0$.

α can be learned based on performance comparison of physics guided and data driven models. Since I can get the range of $\mathrm{E[k]}_{i,j} \in [0, \log_2 N]$ and $D[k]_{i,j} \in [(1+\sqrt{2})/2, D_{\max}]$, I can derive:

$$\log_2 N/D_{\max} \leq \alpha \leq 2(\sqrt{2}-1)\log_2 N. \tag{4.15}$$

α is selected close to $\log_2 N/D_{\max}$ for bias towards the physics guided model, while α is selected close to $2(\sqrt{2}-1)\log_2 N$ bias towards the data driven model. I calculate errors when different α values are applied to the proposed adaptive model with historical data. Then I select the α that gives the minimum error and use this α on the adaptive model applied to the current data. The optimization of the value α can be further obtained by mathematical derivation or optimization methods, which will be investigated in my future work since this section only focuses on the concept of adaptively combining two models.

4.5 Evaluation

In this section, I evaluate my system's ability to address the challenges of **sparse sensing coverage** and **time-variant sensing coverage**. As discussed in Section 1, **sparse sensing coverage** leads to accuracy deterioration while **time-variant sens-**

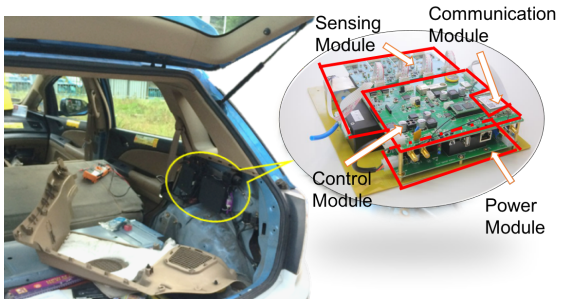


Figure 4.4: The sensing hardware is deployed on the right side of the car's trunk. It is covered up when deployment is finished to prevent it being disturbed during vehicle operation.

ing coverage leads to unstable accuracy. Therefore, I focus on accuracy (average error) and stability (standard deviation of error) to validate system design with data collected from city-scale deployment.

I first introduce how I deploy my system to collect data for evaluation in Section 4.5.1. Then, I derive statistical analysis on sensing coverage in Section 4.5.2, to illustrate the challenges of **sparse sensing coverage** and **time-variant sensing coverage**. Finally, in Section 4.5.3, I validate that my system is capable of dealing with the challenges of **sparse sensing coverage** and **time-variant sensing coverage** with the improvement on accuracy and stability respectively.

4.5.1 Deployment & Experiment Setup

In order to evaluate my approach, I design a mobile sensing hardware platform for city-scale air pollution data collection. I deploy the sensing hardware on 29 electric taxis in an area of 256 km^2 for 14 days.

Sensing Hardware: The sensing hardware is designed for vehicles to collect air pollution data and transmit the data back to the cloud. The sensing hardware includes four parts for different functions: sensing module, power module, communication module and control module [109].

Sensing module is used to collect air pollution information. It includes a GPS sensor to record location and time and concentration of 8 types of gases (CO, CO_2, O_3, NO_2, SO_2, PM_1, $PM_{2.5}$, PM_{10}). The air pollution data is sensed every second. The other three parts work cooperatively to support the sensing module. *Power module* contains a power management integrated circuit to stablize the vehicle's electrical

power and offers stable power support to the sensing module. *Communication module* is responsible for all the transmissions between the sensing hardware and the cloud through both 3G and WiFi. *Control module*, which is mainly composed of a micro-controller, is capable of scheduling sensing, communication tasks in the sensing hardware.

City-Scale Deployment: In order to achieve large scale deployment for system evaluation, I deploy the sensing hardware on 29 electric taxis in the city of Shenzhen for 14 days. Although I do not restrict the motion of the taxis, I define the sensed area to the central business area, which covers 256 km^2, from (113.8668°E **to** 114.1538°E **and** 22.5150°N **to** 22.58868°N). This corresponds to the three most important districts of the city: Luo Hu, Fu Tian and Nan Shan. During the deployment, around 26.3 million data samples are collected and transferred to the cloud.

The sensed area is very sparse within each *time slice* of 1 hour ($1\% - 15\%$ coverage), although around 100,000 data samples are collected. This is because taxi routes cover overlapping areas.

For easy understanding, I transfer the longitude and latitude to kilometers and set the left bottom point as the point of origin in my evaluation. I deploy the sensing hardware in the taxi trunk and an air extracting device pumps air outside the electric taxis into the sensing hardware, as shown in Figure 4.4.

Experiment Setup: Here I outline the experimental configuration, including data description,system parameters, ground truth, performance metric, and baselines.

- **Data Description:** The data collected and transferred to cloud are formatted as: record id, car id, time stamp, longitude, latitude, CO concentration, CO_2 concentration , $PM_{2.5}$ concentration, PM_{10} concentration, SO_2 concentration, NO_2 concentration, O_3 concentration. All the gas concentration is expressed in parts-per million (PPM), which is a commonly used unit.

  For simplicity but without loss of generality, I only focus on deriving CO pollution map. This is because deriving pollution map for other gases are similar and CO is identified as one of six key air pollutants by USA National Ambient Air Quality Standards (NAAQS) as well as Chinese Ministry of Environmental Protection [110]. In addition, CO is a toxic gas. Only 3 - 7 PPM CO leads to 14% increase in the rate of admission in hospitals of non-elderly for asthma and 5 - 6 PPM CO causes significant risk of low birth

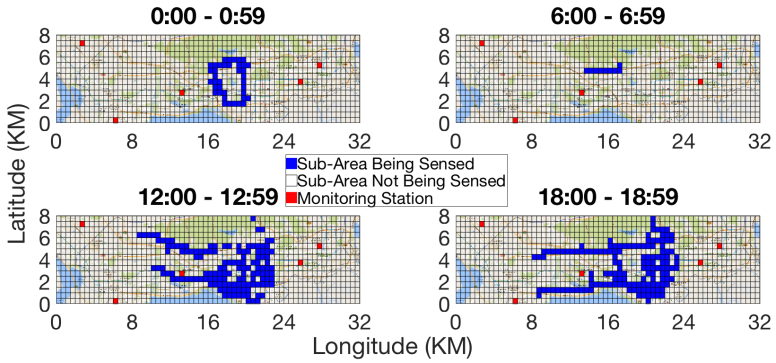


Figure 4.5: The figure shows *sub-areas* being sensed in blue solid square and *sub-areas* not being sensed in blank square within four different *time slices*. The red solid squares are 5 government-run air monitoring stations, which are used as ground truth. The temporal resolution is $500m$ by $500m$. The sensing coverage, i.e. the percentage of *sub-areas* being sensed, changes significantly at different time and locations.

rate if exposed during last trimester[111, 112]. Therefore, CO values are of particular importance for human health. At least less than 1 PPM estimation error should be achieved. Even 1 PPM estimation error on CO may lead to residents' panic or ignorance on air pollution, that threatens their health.

- **Testing Set:** I adopt the measurements from all 5 national air pollution monitoring stations within the experiment area as testing set [113]. These national air pollution monitoring stations publish hourly air pollution data, which is used as testing set of my experiment. As shown in red filled squared in Figure 4.5, the 5 air monitoring stations include: Li Yuan, Hong Hu, Hua Qiao Cheng, Nan You and Xi Xiang. These five monitoring stations are located at the north-west, north-east, center, south-west and south-east of the experiment area. For easy representation, I assign id $1 \sim 5$ to the south-east, north-east, center, south-west and north-west stations respectively.

  However, these 5 monitoring stations are not enough to evaluate my system which claims high resolution and large scale. To address the problem of lacking official air pollution monitoring station data, at each *time slice*, I also adopt data from 40% of the sensed area data as testing set. For example, at time slice k, $40 sub-areas$ are sensed, I randomly select $16 sub-areas$ as testing set, while the rest $sub-areas$ as training data. This guarantees the independence between testing and training sets. Therefore, I have data from both government-

run air pollution monitoring station and my mobile sensing platform as ground truth.

- **System Parameters:** In the air quality application, I applied spatial resolution of $500m$ by $500m$, which correlates to block level scale. I set temporal resolution as 1 hour, which is the same as most government run monitoring stations [92]. The number of particles for Particle Filter is set to 100. This is because for CO estimation, the common CO value ranges from 0 to 5 and 0.1 PPM resolution is good enough. At least 50 particles are needed and I use 100 particle for better accuracy. The α is set as 0.375. The value of α decides the system bias more to the physics guided model of the data-driven model, which can be calculated with the testing results in the past. I calculate errors when different α values are applied to the proposed adaptive model with historical data. Then I select the α that gives the minimum error in the historical data testing, which is 0.375.

- **Performance Metric:** In order to measure the performance of my method, I defined estimation absolute error $E_r[k]_{i,j}$ at *sub-area* (i, j) and *time slice* k as

$$E_r[k]_{i,j} = \left| \mathbf{C}_d[k]_{i,j} - \mathbf{C}_{gt}[k]_{i,j} \right|, \tag{4.16}$$

  where $\mathbf{C}_d[k]_{i,j}$ and $\mathbf{C}_{gt}[k]_{i,j}$ are the derived value and ground truth at *sub-area* (i, j) and *time slice* k respectively. This metric represents deviation between the derived value and ground truth value.

  I use the average value of $E_r[k]_{i,j}$ over *sub-areas* and *time slices* as a measurement of system accuracy and its standard deviation as a measure of the system robustness. Smaller average values and standard deviations represent better accuracy and higher robustness respectively.

- **Baselines:** In order to illustrate how my method, physics guided and adaptive approach (*PGA*), improves performance compared to other existing methods, I adopt the following methods as baselines:

  Pure Physics Guided Model (PHY): This method is the physics guided model I use in my method, which only relies on the temporal and spatial relations from Equation 4.1. By comparing my method with this baseline, I can see the improvement of integrating a data-driven model working with my adaptive

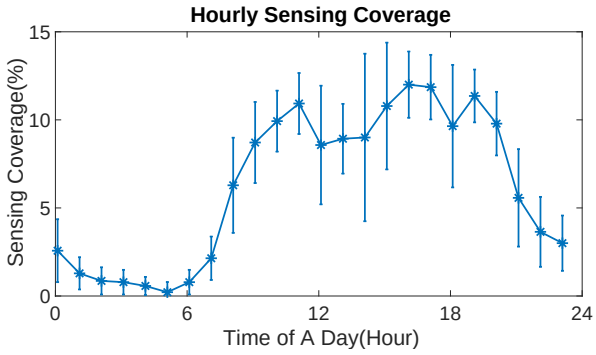


Figure 4.6: This figure shows the averages and standard deviations of hourly sensing coverage from my city-scale deployment.

model fusion scheme.

Pure Data Driven Model (ANN): The artificial neural network(ANN) method is a network composed of simple elements called neurons, which is able to learn complex relationships between inputs and outputs [107]. ANN is the data driven model I use to generate measurement for the adaptive model fusion in my method. By comparing my method with this baseline, I can see the gain from integrating the physics guided model into Particle Filter structure working with the adaptive model fusion scheme.

AirCloud: I derive the air pollution map with the method from a state-of-the-art work AirCloud [87]. AirCloud is a combination of two data-driven models, which adopts Artificial Neural Network for online sensor calibration and Gaussian process to infer the air pollution values at locations where sensors are not available. This work uses similar data source to infer air pollution as my method. Therefore I adopt this state-of-the-art method as one important baseline to compare.

4.5.2 Statistical Data Analysis

To illustrate that sensing coverage is sparse and changes over time and space throughout my deployment, I plotted the hourly sensing coverage at 4 different *time slices* of one day in Figure 4.5. At each *time slice* of an hour, the sensed *sub-area* is very sparse despite a large number of samples. The sensed *sub-areas* at the center of the city is denser because most human activities happen in the center area and taxis

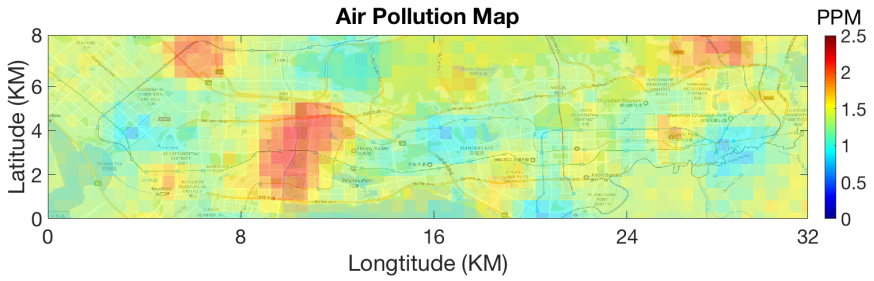


Figure 4.7: The figure shows the hourly air pollution map derived by my physics guided and adaptive approach with the spatial resolution of $500m$ by $500m$.

enter this area more frequently. The sensing coverage is sparser between 6:00am - 6:59am than at the other 3 *time slices*, because people rarely take taxi during this time. This shows how human activities cause **sparse and time-variant sensing coverage**.

For further analysis, I plotted the average and standard deviations of the hourly sensing coverage in Figure 4.6, i.e. the percentage of *sub-areas* being sensed at each hour. The averages and standard deviations are calculated across all the 14 days of the deployment. First, due to overlapping taxi routes, all the average values are less than 12%. Average hourly sensing coverage from 2:00am - 6:00am are around 1%, which corresponds to decreasing taxi mobility due to decreasing human activities during this period. Second, the average values vary from 0.2% - 12% over time. Third, the standard deviations, which show the variances between days, are between 0.5% - 4.8%, which are comparable to the average value range 0.2% - 12%. **These three observations demonstrate the scale of the challenges: sparse and time-variant sensing coverage problems in deriving fine-grained air pollution map.**

4.5.3 System Performance

I compare my method with three baselines to illustrate the advantage of my method design. I evaluate accuracy and stability to validate system design with data collected from city-scale deployment.

Overall Performance: In order to illustrate the overall performance of my method and the three baselines, I plotted the cumulative distribution function (CDF) of average absolute errors in Figure 4.8. To compare how the fmy methods perform

under different sensing coverage conditions, I plotted the results from 6:00am - 6:59am when the sensing coverage is very sparse ($\sim 1\%$) at the top and the results from 18:00am - 18:59am when the sensing coverage is relatively denser ($\sim 10\%$) at the bottom.

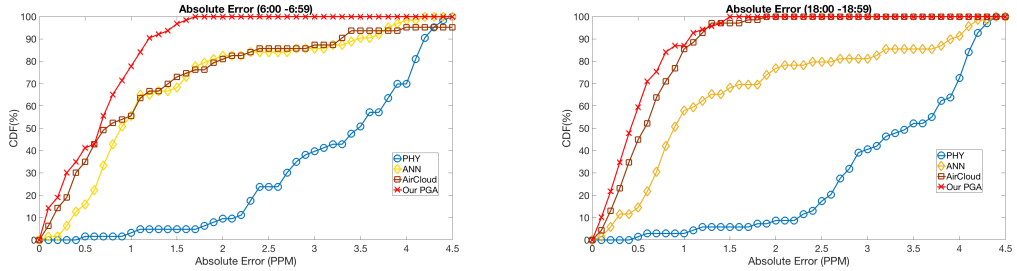
Four methods show consistent performance difference in both sub-figures. My *PGA* performs best, AirCloud ranks second, ANN ranks third and PHY performs worst. PHY's low accuracy is due to its bad online parameter estimation which brings a consistent and accumulating bias. AirCloud is better than ANN since AirCloud is a combination of ANN and Gaussian Process, which takes advantage of both methods (high resolution and high accuracy). My *PGA* performs best since my adaptive scheme is able to choose model with better accuracy, which limits the error. When data is sparse, *PGA* is biased to the physics guided model and when data is dense, *PGA* is biased to the data driven model.

Given the absolute error, AirCloud show lower percentages during 6:00am - 6:59am than 18:00am - 18:59am, since performance of AirCloud rely on sensing coverage. In contrast, my method does not show too much difference between two sub-figures. The consistency comes from adopting physics guided model which preserves the air pollution temporal and spatial relationship.

Figure 4.8(a) shows the real-world situation of very sparse sensing coverage. Given absolute error $< 1PPM$, PHY, ANN and AirCloud only gets 0%, 56% and 56% respectively, while my method gets 89%. Furthermore, all of my estimation are limited to $1.6PPM$. In contrast, all three baselines have absolute error $> 3.5PPM$.

I believe that the improvement comes from the integrating the physics guided model. AirCloud combines two data-driven models, which means that the limitation of the data-driven model when the data is sparse cannot be relieved. On the other hand, my *PGA* introduces the physics guided model to handle **sparse sensing coverage** and **time-variant sensing coverage**, which shows a $4\times$ reduction in the error $< 1PPM$ compared to the AirCloud.

I observe that my deployment is focused on the busy areas and the center area of a city, which sometimes can achieve high sensing coverage as Figure 4.8(b). However, for even large scale deployment, hourly 10% sensing coverage is really difficult to achieve and 1% sensing coverage is a quite common number. At this condition, my *PGA* shows significant improvement over the state-of-the-art method AirCloud, as show in Figure 4.8(a).



(a) Cumulative Distribution Function (CDF) of Absolute Errors during 6:00-6:59.

(b) Cumulative Distribution Function (CDF) of Absolute Errors during 18:00-18:59.

Figure 4.8: This figure shows cumulative distribution function (CDF) of absolute errors, using my method and three baselines during 6:00-6:59 when sensing coverage is very sparse ($\sim 1\%$) and during 18:00-18:59 when sensing coverage is relatively denser($\sim 10\%$). At both time, my method consistently shows performance advantages over three baselines.

Performance VS. Time: In order to check how four methods perform over time, I plot daily average and standard deviations of relative error at the top and bottom in Figure 4.9. Each average and standard deviation value is obtained from 24 hours of the day and all ground truth locations, which involves both temporal and spatial statistics.

First, my PGA achieves $0.30PPM - 0.59PPM$ absolute error, which are lower than PHY and ANN. To be specific, my method has up to 13.9× and up to 4.7× reduction on average absolute errors compared to PHY and ANN respectively. This shows that my adaptive model fusion, working with Particle Filter structure, is able to consistently determine whether physics guided model (PHY) or data driven model (ANN) is better on different days, which have different traffic patterns. The successful decision based on quality comparison between two models helps the system optimize both accuracy and robustness.

Second, ANN shows $0.68PPM - 1.04PPM$ absolute error standard deviation, which are up to 7.3× higher than my PGA and up to 6.6× PHY. This is because the accuracy of ANN depends on the sensing coverage, which is highly time-variant. In contrast, both PHY and my method shows much lower standard deviation, i.e. higher robustness. This proves the stability of the physics guided model within a 24 hour period. My system also keeps the stability by integrating the physics guided model.

Finally, compared to AirCloud, my method shows increased performance on both

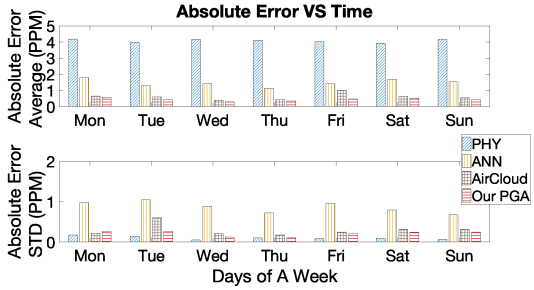


Figure 4.9: This figure shows the absolute errors on different days of a week from my method and three baselines. The average values are shown at the top figure while the standard deviations are shown at the bottom. Compared to three baselines, my method consistently shows lower average absolute errors on different days of a week. Besides, my method shows a lower or similar standard deviation, i.e. higher robustness.

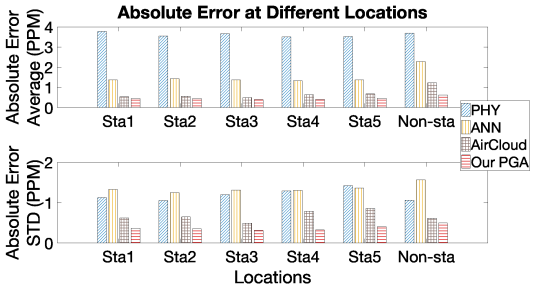


Figure 4.10: This figure shows average and standard deviation of absolute errors at 5 monitoring stations and other non-station areas. My method consistently has lower values on averages and standard deviations at all locations, which means higher accuracy and robustness.

average and standard deviation of absolute errors. My method shows up to 2.1× and 2.2× reduction on average and standard deviation compared to AirCloud, which is a two stage method combining two data driven model, Artificial Neural Network and Gaussian Process. This shows my *PGA*'s advantage over the state-of-the-art method. The advantage comes from the combination of one physics guided model and one data driven model.

Performance VS. Location: In order to evaluate how the four methods perform over locations, I plot average and standard deviations of absolute error 5 government-run air pollution monitoring stations and other non-station locations in Figure 4.10. All averages and standard deviations are obtained from 14 days by 24 hours.

My *PGA* method shows average of $0.39PPM - 0.62PPM$ and standard deviation of $0.31PPM - 0.50PPM$, which have consistent advantages over three baselines at all locations. Specifically, my method shows up to 9.2×, 3.7× and 2.0× reduction on average errors, and up to 3.9×, 4.2× and 2.4× improvement on standard deviation, compared to PHY, ANN and AirCloud respectively. This shows that my adaptive scheme is able to consistently limit the absolute errors at different locations, which leads to higher accuracy and robustness.

I observe that PHY standard deviations in Figure 4.10 are higher than that in Figure 4.9. This is because the standard deviations in Figure 4.10 are calculated over different days while the standard deviations in Figure 4.9 are calculated within one day. PHY has good short-term robustness, but the robustness drops in the long-term.

Performance VS. Sensing Coverage: In order to see how sensing coverage affects system performance, I plotted results of four methods under different sensing coverage conditions in Figure 4.11. I select the time periods when the system collects same amount of data, i.e. achieves same sensing coverage, then calculate the average and standard deviation for absolute errors. The center of the bars show the average absolute errors while the length of the bars show the standard deviation of absolute errors.

When sensing coverage increases from 1% to 10%, my *PGA*'s average absolute error drops from $0.59PPM$ to $0.11PPM$ and standard deviation drops from $0.26PPM$ to $0.11PPM$. Similar decreasing trends are also observed for other three baselines. This shows that sensing coverage affects the system accuracy and reliability for all methods.

ANN and AirCloud show large disparities on average and between highest sensing coverage and lowest sensing coverage ($0.78PPM$ for ANN and $0.54PPM$ for AirCloud). The large disparities also show in standard deviation for ANN ($0.44PPM$) and AirCloud ($0.53PPM$). In constrast, disparity is much smaller for PHY and *PGA* on both average ($0.2PPM$ for *PGA* and $0.2PPM$ for PHY) and standard deviation ($0.15PPM$ for *PGA* and $0.29PPM$ for PHY). This is because that both ANN and AirCloud are data-driven methods, whose performance highly depends on sensing coverage. In contrast, PHY does not rely on sensing coverage that much. *PGA*, as a fusion of physics guided and data driven method, with the help of physics guided model and adaptive scheme, has the capability to resist the performance variation caused by sensing coverage variation.

My method has better accuracy and stability than three baselines under different sensing coverage. Specifically, my method shows up to 10.5×, 5.0× and 1.8× accuracy improvement over PHY, ANN and AirCloud respectively, as well as 3.0×, 8.0× and 3.7× stability improvement over PHY, ANN and AirCloud. AirCloud gets $0.46PPM$ and $0.37PPM$ for average and standard deviation when sensing coverage is 10%. My *PGA* achieves similar performance when sensing coverage is 2%. This means

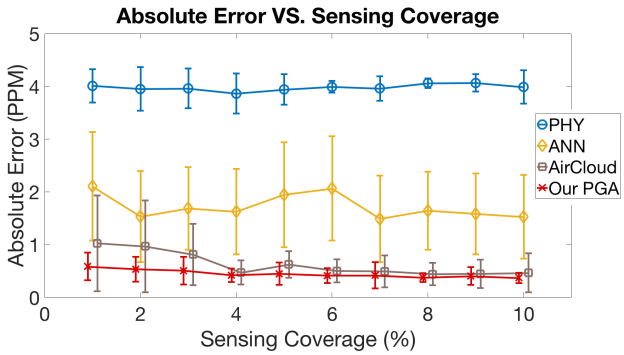


Figure 4.11: The figure shows the average and standard deviation of absolute error of my method and three baselines at different sensing coverage conditions. At any sensing coverage, my method maintains increased performance over the baselines. The performance from all four methods deteriorate when sensing coverage becomes sparse, but my method shows smaller differences in average absolute error between highest and lowest sensing coverage.

that to achieve similar performance, my *PGA* requires 5× less sensor deployment than AirClould and even less than other methods. The advantage comes from the adaptive scheme's ability to biase to the physics guided model and the data driven model when sensing coverage is low and high respectively. This adaptation enables the system to limit the error and error variation.

This section evaluates the accuracy and stability of my system and three baselines, considering overall performance, time variance, location variance and coverage variance. The results show that PHY has very large errors, since the errors of this model accumulate over time. At *time slice* k, before estimating the air pollution, the key parameters $U[k]$ need to be estimated first. The quality of estimating $U[k]$ depends on the quality of air pollution estimation before k. As a result, the error of PHY is accumulative. In contrast, the performances of ANN and AirCloud only rely on the current data collection quality, which leads to high variance, i.e. low stability. Finally, my *PGA* combines the physics guided model and data-driven model together with an adaptive scheme. The physics guided model helps to lower the variance of estimation with its temporal and spatial physical constraints, while the data-driven model helps to prevent the error accumulation problem from PHY.

To summarize my system evaluation, I deploy self-designed air pollution sensing hardware on 29 electric taxis in an area of $256km^2$ for 14 days. I compare my system with a pure physics guided model (PHY), a pure data driven model (ANN)

and a state-of-the-art method AirCloud in terms of estimation absolute error. The evaluation covers overall performance, performance at different times, performance at different locations and performance with different sensing coverage. my *PGA* adaptively biases to the physics guided model when and where sensing coverage is sparse and biases the data driven model when and where sensing coverage is dense. The evaluation results show my system achieves up to 13.9×, 5.0× and 2.1× reduction of average error and up to 3.9×, 8.0× and 3.7× improvement on robustness compared to the pure physics guided model, ANN and AirCloud respectively. To achieve similar performance, my *PGA* requires at least 5× less sensor deployment than other approaches.

4.6 Related Work

This section discusses work related to this section. I first discuss different sensing platforms for air pollution data collection. Then different methods for deriving, estimating and inferring air pollution maps are discussed.

4.6.1 Air Pollution Sensing Platforms

Sensing platforms for air pollution can be categorized into three classes. Satellite-based [114, 115, 116] monitoring station based on [92, 93] and mobile based [87, 88, 94]. Satellite-based and station-based approaches are expensive to build and maintain. As a result, these methods tend to have very sparse sensing coverage. The low cost of mobile sensing devices on vehicles and drones allows large scale deployment and high mobility. This paper is also based on data collected with a mobile sensing platform. Prior approaches are similar to my Gaussian Process baseline, and are sensitive to data sparsity. Thus, high device penetration is needed.

4.6.2 Methods to Derive Pollution Map

Two major classical methods are commonly used to derive an air pollution map. One is based on physical characteristics of air dispersion [97, 117], while another is a pure data-driven method, such as interpolation and machine learning [87, 92].

Using physical relations to derive air pollution information has been studied

considerably in both sensing domains and environmental monitoring domains [97, 117]. These models are mostly used to conduct numerical simulations on air dispersion with known initial states and pollution sources [90, 91]. There are also works that use a model to estimate pollution sources, which mostly require a known source's location, which is often not possible to obtain in a city [118]. They can achieve high resolution based on the continuity of physical laws. However, the accuracy of these model approaches relies on known parameters which are difficult to obtain a priori, and often vary over time in non-simulated systems similar to the pure physics guided model baseline. The model errors as well as the parameter estimation errors cause inaccuracy and instability of these methods.

There are several data driven models focusing on deriving an air pollution map with data collected by mobile sensing platforms [119, 120, 121, 122]. U-Air and Air Cloud [87, 92] infer real-time and fine-grained air quality information throughout a city using a machine learning approach. U-air requires a lot of information sources, including meteorology, traffic flow, human mobility, structure of road networks and points of interest. The information is difficult to obtain and contains a lot noise. The large noise could lead to performance deterioration. Yidan Hu etc. [94] proposed a 3D probabilistic air pollution concentration estimated method to infer $PM_{2.5}$ based on Brownian motion. These methods rely on air quality data as reported by existing monitor stations or mobile sensing platforms as well as a variety of data sources observed in the city such as meteorology, traffic flow, human mobility, the structure of road networks, and points of interest. These approaches can achieve high accuracy given enough data. However, the performances of these methods are highly reliant on the amount and coverage of the data collections.

Previous methods do not consider the highly time-variant changing sensing coverage. In addition, these methods either do not combine models or combine two data driven models. The advantage of a physics guided model is not considered. As a result, the advantages of these two classical methods are not well exploited.

4.7 Conclusion

This chapter presents *PGA*, a physics guided and adaptive approach to derive a fine-grained ($500m$ by $500m$ by $1hour$) air pollution map. The system is designed based

on a Particle Filter structure since it offers the ability to combine the advantages of a physics guided model and data driven model. 1) To address the challenge of sparse sensing coverage, the system adopts the temporal and spatial relations of air dispersion from a physics guided model to extract more information for reconstruction (ST-PHY). The physics guided model is reformulated as the state evolution estimation model in the Particle Filter structure. 2) To address the dynamic sensing coverage challenge, the system adaptively trusts either the result directly from the physics guided model or the result after correction through measurements generated from the data driven model. The choice is made based on quality comparison of the estimate by the physics guided model and the generated measurement by the data driven model to optimize performance for each *sub-area* and each *time slice* (ST-SYS). To evaluate the system, I deployed 29 taxis in the city of Shenzhen for 14 days, which collected around 26.3 million data samples. The evaluation results show my system achieves up to 13.9×, 5.0× and 2.1× reduction of average error and up to 3.9×, 8.0× and 3.7× improvement on robustness compared to the pure physics guided model, ANN and AirCloud respectively. To achieve similar performance, my *PGA* requires at least 5× less sensor deployment than other approaches. All these validate the effectiveness of applying spatiotemporal relationship on the large scale mobile CPS for physical field estimation.

Chapter 5

Actuation Planning for Controllable Large Scale Mobile CPS

This chapter introduces how to utilize spatiotemporal relationships to address the challenge of inaccurate system status estimation for route planning in the controllable system (MAV autonomous navigation), as described in Chapter 1. Compared to two uncontrollable systems in Chapter 3 and 4, the controllable system has the ability to plan routes for mobile devices. By dispatching the mobile devices to key locations, the system is able to collect more informative data to understand the physical process better. However, route planning in the controllable large scale mobile CPS is not easy since route planning depends on both the estimation of system status and planning strategy. Due to the constraints of cost, weight, size etc., each device only has limited capabilities of sensing, computing, and communication, which makes system status estimation challenging. In addition, since the system cannot always ensure accurate estimation of system status, route planning has to adapt to the quality of the estimation. To address the challenges, the *spatiotemporal relationship analysis layer* improves the system status estimation of individual nodes at their trajectory intersections according to similar radio signatures (ST-PHY-SYS). Based on the quality of system status estimation at different times and locations (ST-SYS), the system adaptively plans the routes .

The system overcomes limitations of individual devices by improving the system

status estimation of individual nodes at their trajectory intersection according to similar radio signatures (ST-PHY-SYS). The system utilizes the quality of system status estimation at different times and locations (ST-SYS) to guide the adaptive navigation.

I first motivate the problem in Section 5.1. Then section 5.2 gives a high level overview of the architecture and operation of the system. Section 5.3 gives a detailed technical description of the various algorithms presented in the paper. Section 5.4 evaluates and analyzes the system through extensive simulations and validates assumptions through MAV testbed experiments. In Section 5.4.7, I describe related work and discuss the state-of-the-art infrastructure-free navigation techniques in context of MAV swarm deployment. Finally, I draw conclusions and summarize my contributions in Section 5.5.

5.1 Problem Overview

In many hostile, dangerous, or otherwise inaccessible environments (such as urban search and rescue, environmental monitoring, surveillance, etc.), situational awareness is needed. However, in these dangerous scenarios, manual deployment of sensors is often not feasible.

In such scenarios, autonomously navigating MAV swarms to a set of goal locations, in accordance with the needs of domain experts, can provide significant benefit. Further, utilizing a large number of low-cost, low-complexity mobile sensor nodes, as opposed to using a limited number of sophisticated robots, can be more cost effective and provide increased robustness through redundancy. In addition, small lightweight mobile sensor nodes provide greater safety as the effects of their collisions with the objects or persons in the indoor environment are inconsequential.

MAV swarms are an emerging class of networked mobile systems with widespread applications in such domains. These swarms consist of miniature aerial sensor nodes with limited individual sensing, computing and communication capabilities [123, 124]. Initial work in the operation of MAVs has focused on outdoor or highly instrumented environments that rely on external sensors to control individual devices [125, 126]. However, such centralized sensing approaches are hampered in indoor environments by obstructions (walls, furniture, etc.). At the same time, reliance on sensing

infrastructure implies the requirement of a large deployment of support sensors covering all the locations that a MAV may visit [127]. Thus these approaches are only applicable in pre-surveyed locations.

This section presents DrunkWalk, a technique for cooperative and adaptive navigation of swarms of micro-aerial sensors in environments not formerly preconditioned for operation. The key focus behind this networked MAV swarm research is to rely on collaboration to overcome limitations of individual nodes and efficiently achieve system-wide sensing objectives. In addition, I also extend my algorithm to a heterogeneous swarm structure, where *advanced MAVs* (with high accuracy sensors) and *basic MAVs* (with low accuracy sensors work) collaboratively in a team.

In DrunkWalk, the MAV swarm self-establishes a temporary infrastructure of a few landed MAV's acting as radio beacons. Using radio signature or fingerprints from beacon nodes, the algorithm detects intersections in trajectories of exploring mobile MAV nodes. The algorithm combines noisy dead-reckoning measurements from multiple MAVs at the detected intersections to improve the accuracy of the MAVs' location estimates. Most importantly, to adaptively plan trajectories of MAV nodes according to the certainty of their location estimations – directing movement to improve location estimates when certainty is low (exploration mode), and directing MAV to the goal location when certainty of location estimates is high (navigation mode). The adaptive strategy enables DrunkWalk to improve the location estimation accuracy and success rate of navigation under given time and accuracy constraints. In addition, DrunkWalk also considers the adaptation to a heterogeneous swarm structure. By assigning different roles to MAVs through different settings according to different sensing capabilities, DrunkWalk makes *advanced MAVs* stay in exploration mode more to search the environment for the whole team.

The main contributions of this section are:

- An adaptive planning algorithm for navigation that enables the swarm to collaboratively achieve up to 6× reduction in location estimation errors, and as much as 3× improvement in navigation success rate under the given time and accuracy constraints.
- A planning algorithm that determines the quality of location estimations and uses it to adaptively plan node motion.

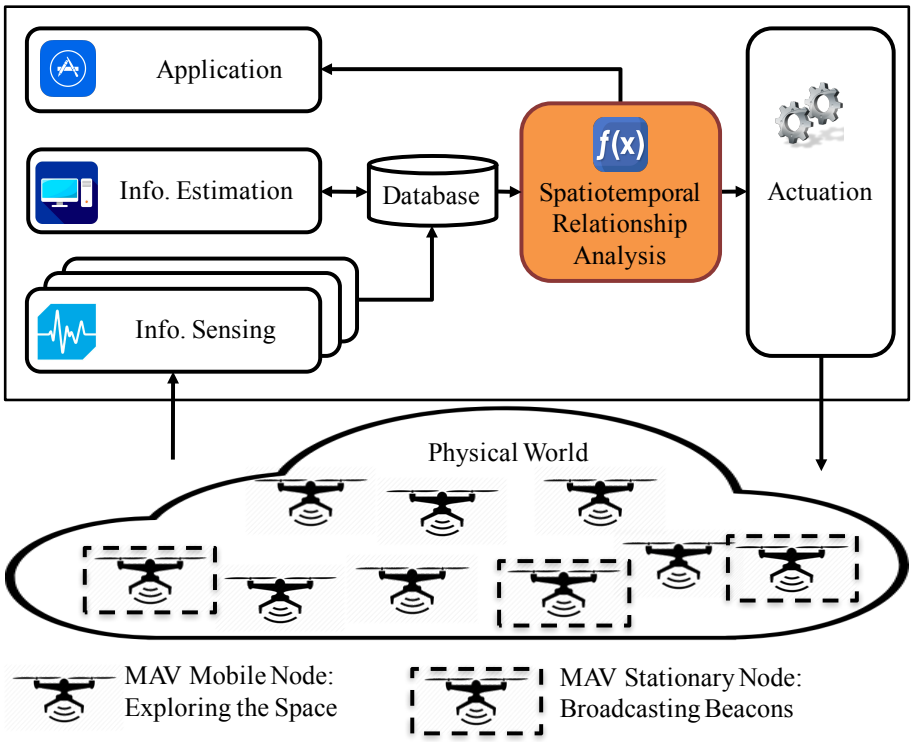


Figure 5.1: This figure shows the architecture of spatiotemporal relationships aided large scale controllable drone based CPS for estimating system status (MAV swarm localization) and route planning (MAV swarm navigation).

- Real MAV testbed experiments and large scale physical feature based simulations using radio signatures collected from the physical world and empirically determined sensor noise models validating my assumptions. Drone swarms with both homogeneous and heterogeneous structures are tested.

5.2 System Overview

Potential MAV swarm sensing applications require mobile sensors to autonomously navigate to desired locations in operating environments with no localization infrastructure. In this section, I address the problem of how a network of mobile sensors can be navigated to pre-determined positions under time and accuracy constraints.

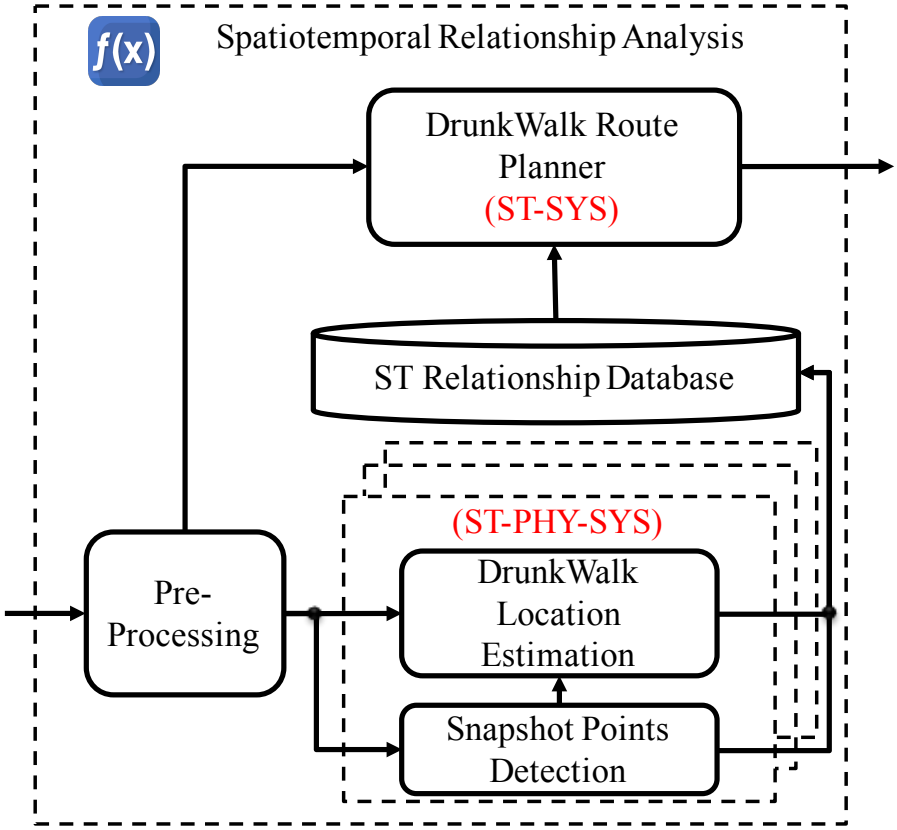


Figure 5.2: This figure shows the architecture of the *spatiotemporal relationship analysis layer*, which 1) improves the system status estimation (MAV swarm localization) of individual nodes at their trajectory intersections according to similar radio signatures (ST-PHY-SYS), and 2) adaptively plans the routes (MAV swarm navigation) based on the quality of system status estimation at different times and locations (ST-SYS).

5.2.1 Operation & Architecture

The system begins operation with a swarm of MAV's being introduced into the operating environment. I make the assumption that a coarse map of the building is available and can be utilized by domain experts to pre-determine suitable placement of sensors. This is a valid assumption in most scenarios, as emergency response personnel have access to the rough floor-plans of buildings through city registries, and thanks to increased availability of indoor maps tailored to location based services (e.g., indoor Google maps).

The proposed system has 3 major operational phases: setup, estimation and planning (the latter two proceed in conjunction):

- **Setup:** The system autonomously establishes a transient infrastructure of stationary MAV nodes acting as wireless beacons. These nodes land upon being introduced into the area and remain stationary during the process. The objective of the stationary nodes is to enable mobile MAV nodes to obtain radio signatures or fingerprints of locations traversed on their paths. These nodes use a simple dispersion algorithm [128, 129] that lets them spread out in the environment *without any estimation of their location.*
- **Location Estimation:** The system then desires to estimate the locations of nodes in order to guide them to their goal locations. To realize this, the system first uses dead reckoning sensors such as an optical flow velocity sensor and magnetometer (in my test MAV platform) to get a rough estimate of the motion path of mobile nodes. Second, the system uses radio fingerprints, collected by mobile nodes from the self-established wireless beacons, to determine *snapshot points*, i.e. location points that were previously visited by other nodes or by itself. Finally, the system uses the snapshot points to combine location estimates from multiple nodes and collaboratively improve location estimations of the entire swarm.
- **Actuation Path Planning:** Having estimated locations, the system plans paths for each node that 1) leads to subsequent goal positions and 2) improves location estimation accuracy. The quality of the planned path depends greatly on the accuracy of the initial location estimate of nodes. A bad location estimate will render any attempt to plan a deterministic path useless – *when the nodes don't know where they are, they cannot plan a correct path to their destination.*

  my system thus considers the quality of location estimation in planning node paths. The path planner commands node movements such that they increase the number of snapshot points and potentially improve location estimates when the quality of their estimates is likely to be low. On the other hand, when the location estimates are likely to be more accurate, the planner uses the map to direct them to their designated goal locations.

Figure 5.1 shows the architecture of the system. Through a dispersion algorithm, the system deploys **Stationary MAV Nodes** that act as wireless beacons. **Mobile MAV Nodes** explore and obtain dead-reckoning measurements from their on-board

sensors and radio RF-signatures from the stationary beacons. The *information sensing layer* collects radio RF-signatures and the navigation sensor readings, which are stored in the *database*. The *information estimation layer* calculates the dead-reckoning measurements based on the navigation sensor readings in the *database* and stores the results back to the *database*. The *spatiotemporal relationship analysis layer* utilizes the radio RF-signature and the dead-reckoning measurements from the *database* to conduct the DrunkWalk algorithm and output command to the *device actuation layer* for MAV swarm navigation.

Figure 5.2 shows the details of the *spatiotemporal relationship analysis layer*. The *spatiotemporal relationship analysis layer* utilizes radio RF-signatures to determine snapshot points in node paths and apply corrections to their dead-reckoning estimates (DrunkWalk Estimation). The corrected location estimates and a coarse map of the environment are stored in the *spatiotemporal relationship database*, which are used to command the subsequent movements of MAV nodes (DrunkWalk Planner). In addition, in order to adapt to the heterogeneous MAV swarm structure, DrunkWalk assigns different settings to different MAVs based on their sensing capabilities, which causes MAVs with high accuracy stay in exploration mode more to search the environment for the whole team.

5.2.2 Improving Location Estimation Through Swarms

The core idea behind my estimation approach is to use the relatively large number of mobile sensors in the swarm to collaboratively reduce the error. This is achieved by detecting when nodes move over the same space in the environment and combining their individual location estimations at these points. Errors in dead reckoning measurements are mainly due to noise in inertial sensors that are independent across nodes and time [130]. Thus, combining estimates from multiple nodes and propagating corrections to them improves their location estimations. Figure 5.3 illustrates the on-line process of determining snapshot point from radio measurements.

Determining Snapshot Points: The location estimation requires a node to be able to determine when it visits a location previously visited by itself or by another nodes - *a snapshot*. The snapshot point provides the opportunity to combine estimations from multiple independent mobile nodes and improve location estimations.

The system determines a snapshot point using radio fingerprints collected by mobile nodes from the self-established beacon nodes. The radio fingerprints are collected in an *online* fashion, i.e., the nodes discover fingerprints as they explore the space. These fingerprints are sent to the Base and matched with a database of previously discovered signatures. If the signature matches an existing signature in the database (decided by a cosine distance and a threshold), the point is classified as a snapshot point and a correction can be applied to the current location estimation. If the signature does not match any existing signatures, it is added to the database as a new entry.

Combining Estimates at Snapshot Points: The process of combining location estimations must be performed carefully. The naive approach would be to take the average of all location estimates for a particular snapshot point. However, this approach does not consider the nature of the underlying distribution of noise in location estimations that often does not follow a normal distribution especially in indoor environments.

Combining estimations is a chicken and egg problem that requires a snapshot point to estimate and update its own location from visiting mobile nodes, and subsequently, use the updated location to correct the estimates of the visiting mobile nodes.

To achieve this, I employ a particle filter based approach. A particle filter [100] is a Bayesian estimation method to estimate system state based on multiple noisy sensor measurements. I use a particle filter to track the position and orientation of each mobile node. Similarly, I use a particle filter to track the position of each snapshot point as it is discovered and visited by the MAV nodes. Every visit to a snapshot point by a mobile node results in the the mobile node *correcting* the estimation of the particles of the snapshot point, which in turn *corrects* the estimations of the particles of the mobile node. The various estimation algorithms are described in detail in Section 5.3.

5.2.3 Adaptive Path Planning

I described how a snapshot point between the paths of nodes can be utilized to improve location estimates. Planning paths is thus the second chicken and egg problem encountered in navigation. Better location estimates are needed by nodes

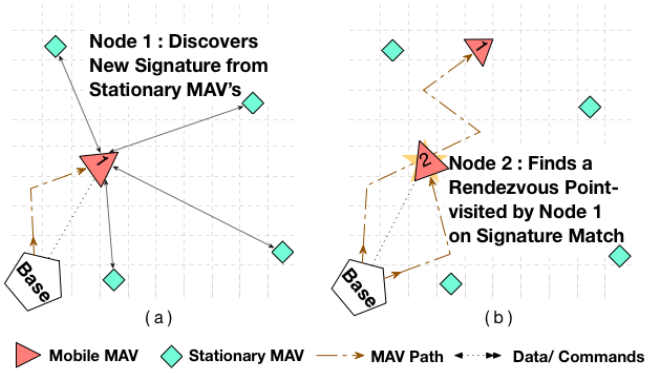


Figure 5.3: The figure shows the process of determining snapshot points. (a) Node 1 moves and obtains a radio signature from stationary MAV. This is entered into the the Base Signature DB as new signature. (b) When Node 2 visits the same location, its collected radio signature matches existing signature and a correction can be performed at the Base.

to navigate to predetermined regions quickly. However, at the same time, achieving better location estimations may require nodes to take detours (to find snapshot points) costing time and energy. The planning component of my system seeks to make a suitable trade-off between these aspects of navigation.

DrunkWalk: In order to reach the goal regions, I use an indoor layout with known locations of walls and doors of the environment. It should be noted that the algorithm does not require high quality maps with information of the position of obstacles. Such rough maps are generally available or easy to obtain in most application scenarios.

The rough map enables us to bias the direction of node movement towards predetermined goal regions, if the current location of the node in the map can be reasonably determined. However, due to noisy sensors, the location of individual nodes cannot always be estimated correctly, which makes it difficult to consistently plan correct paths. The system attempts to solve this by operating in two modes:

- **Exploration:** In this mode, the MAV node attempts to seek snapshot points that can potentially improve the location estimates of the MAV node. This is executed when the quality of location estimations (determined by the entropy of the tracking particle filter distribution) is low.
- **Navigation:** In this mode, the MAV node attempts to follow the direction of the bias from the graph using the estimated location from the DrunkWalk

algorithm. This is executed when the quality of location estimates is high.

It is easy to see that the performance of the navigation step depends on the outcome of exploration step. However, the exploration step requires extra use of resources that increases the time of navigation. Therefore, the DrunkWalk algorithm seeks to optimize this trade-off by adaptively switching between these two modes.

Adaptation to Heterogeneous MAV Swarm: DrunkWalk is originally designed for homogeneous MAVs swarm, where all the MAVs have the same sensing capabilities. However, heterogeneous MAV swarms are becoming more and more popular, where *advanced MAVs* (with more accurate sensors) and *basic MAVs* (with less accurate sensors) coexist. With such heterogeneous structure, *advanced MAVs* should play more important roles than *basic MAVs*. In such a case, I assign different settings to *advanced MAVs* and *basic MAVs*. The intuition here is to make *advanced MAVs* stay in the **Exploration** mode more to help the team search the environment. This is because *advanced MAVs* are equipped with more accurate sensors, so the estimates based on their sensing data are more accurate. Therefore, having *advanced MAVs* search the environment more improves the estimate and navigation accuracy for the whole team.

5.3 Algorithm Design

This section provides a detailed description of the major components of the proposed system. First, this section describes how the location and orientation of the MAVs and the positions of the signatures are estimated over time using a set of particle filters. A separate particle filter is associated to each MAV in the team and to each RF-signature being localized in space. Therefore, particles estimating the position and orientation of the MAVs include the components c_x, c_y, c_ϕ, whereas particles estimating the location of the signatures include components s_x, s_y for the position. As described in Section 5.2, a base station exchanges information with the MAVs (commands and measurements) and maintains a database of known signatures (see Figure 5.2). Due to the limited on-board computational power on the MAV, my current implementation performs all computations in the base.

5.3.1 Particle Filter Background

A particle filter is a Bayesian estimation method using a finite number of elements (so called *particles*) to represent a non-parametric probability density. It was introduced in the fifties [99] and became popular in robotics in the last two decades [100].

As a specific implementation of a more general recursive Bayes filter under the Markov assumption, it requires assumption of availability of two probabilistic models, namely the state evolution model (often called motion model in mobile or robotic applications) and the measurement model. Assuming the unknown state to be estimated at time t is indicated by x_t, the state evolution model provides

$$p(x_t|x_{t-1}, u_t) \tag{5.1}$$

where u_t is the known command given to the system at time t. The measurement model, instead, is given by

$$p(z_t|x_t) \tag{5.2}$$

where z_t is the measurement at time t. Due to the Markov assumption, x_t is conditionally independent from x_k with $k < t-1$ once x_{t-1} is known. Similarly, given x_t, the measurement z_t is conditionally independent from any other variable. Note that one does not need to commit to specific distributions in Eq. 5.1 and Eq. 5.2, e.g., they do not have to be Gaussian distributions. The generic algorithm to propagate a posterior using a particle filter is given in Algorithm 2, where I mostly follow the notation presented in [100]. The algorithm starts with a set of M particles $\mathcal{X}$ estimating the posterior of x_{t-1}, i.e., the state x at time $t-1$. Given the latest command u_t and measurement z_t, it produces a new set of M particles providing an updated posterior estimate for x at time t. The ith particle in $\mathcal{X}_t$, $x_t^{[i]}$, represents the ith possible hypothesis about the state at time t. Algorithm 2 shows the generic particle filter algorithm. The first *for* loop creates a new set of M particles sampling the motion model from the set of existing particles, while the second *for* loop implements the so-called *importance resampling*. The set of particles provides a discrete approximation for the posterior.

Algorithm 2: Generic particle filter algorithm

Data: $\mathcal{X}_{t-1}, u_t, z_t$
Result: $\mathcal{X}_t$
1 $\mathcal{X} \leftarrow \emptyset$;
2 $\mathcal{X}_t \leftarrow \emptyset$;
3 **for** $i \leftarrow 1$ **to** M **do**
4 $\quad x_t^{[i]} \leftarrow$ sample $\sim p(x_t|x_{t-1}^{[i]}, u_t)$;
5 $\quad w_t^{[i]} \leftarrow p(z_t|x_t^{[i]})$;
6 $\quad \mathcal{X} \leftarrow \mathcal{X} \cup \{< x_t^{[i]}, w_t^{[i]} >\}$;
7 **end**
8 **for** $i \leftarrow 1$ **to** M **do**
9 $\quad$ draw j with probability $\alpha\ w_t^{[j]}$;
10 $\quad \mathcal{X}_t \leftarrow \mathcal{X}_t \cup \{x_t^{[j]}\}$;
11 **end**

5.3.2 MAV Location Tracking

In this subsection, I show how the generic particle filter estimator can be specialized to estimate the location of the MAVs. To reduce the computational complexity, rather than implementing a centralized particle filter jointly estimating the location of all the MAVs, I associate a particle filter to each MAV. Assuming there are N_M MAVs involved in the navigation task, the system then creates and updates N_M particle filters. Each filter is initialized with $M = 100$ particles uniformly distributed in the area. All computations take place on the base station.

Prediction from Motion Models: For the prediction step, it is necessary to use a generative law to implement the particle creation in line 4 of Algorithm 2. To this end, I use equations similar to the ones given in [129]. Let the command at time t be $u_t = (v_t, \omega_t)$, where v_t is the translational velocity and ω_t is the rotational velocity. Note that the control system always generates commands in which only one of the two components is different from 0, i.e., the MAV either translates or rotates, but does not make both movements at the same time. Then, a new particle is generated as

$$\begin{pmatrix} c_x \\ c_y \\ c_\phi \end{pmatrix}_t^{[i]} = \begin{pmatrix} c_x \\ c_y \\ c_\phi \end{pmatrix}_{t-1}^{[i]} + \delta t \begin{pmatrix} v_t \cos(c_{\phi t-1}^{[i]}) \\ v_t \sin(c_{\phi t-1}^{[i]}) \\ \omega_t \end{pmatrix} \tag{5.3}$$

where δt is the time interval between two commands. The correctness of the equation follows form the assumption that only one of v_t and ω_t can be different from 0. Noise is added to the translational and rotational velocities as per the empirically obtained actuation noise models $p(n_v)$ and $p(n_\omega)$ from my test MAV platform, but can be specified as per the specific sensor or MAV platform used. Thus, $v_t^{[i]}$ and $\omega_t^{[i]}$ are obtained as:

$$v_t^{[i]} = v_t + n_v^{[i]} \ , \ n_v^{[i]} \ is \ drawn \ from \ p(n_v) \tag{5.4}$$

$$\omega_t^{[i]} = \omega_t + n_\omega^{[i]} \ , \ n_\omega^{[i]} \ is \ drawn \ from \ p(n_\omega) \tag{5.5}$$

where v_t and ω_t are the nominal commands. In my simulations, according to [129], $p(n_v)$ and $p(n_\omega)$ are specified as normal distributions with $\mu = 0$ and σ is expressed as a percentage of the value of v_t or ω_t.

Correction from Measurements: The correction step hinges on the weights assigned to the particles (line 5 in Algorithm 2). Each MAV is equipped with a magnetometer sensor returning a measurement for its heading. Moreover, the RF-signature snapshot provides another measurement. These two measurements are asynchronous in the sense that, while the on-board heading sensor can be queried after each command is executed, signature matching occurs only when revisiting a location associated with a known signature. In the following, I therefore separately describe how the two different weights are computed, given that they are generated and used (via resampling) in separate stages.

The heading measurement is straightforward to integrate. According to former experimental measures [131], the nominal heading returned by the sensor is affected by Gaussian noise with a known variance σ^2 ($\sigma = 40$ degrees to be precise). Therefore, for the heading weight I set

$$p(z_t|x_t) = f_{\mathcal{N},\sigma^2}(z_t - c_{\phi t}^{[i]}) \tag{5.6}$$

where $f_{\mathcal{N},\sigma^2}$ is the density probability of a Gaussian with 0 mean and variance σ^2, and the argument $z_t - c_{\phi t}^{[i]}$ is normalized to account for the 2π period.

The process is substantially different for RF-signature snapshot points. In this case, rather than computing $p(z_t|x_t)$, I determine $w_t^{[i]}$ through a two step process. 1) When a signature is measured, the first step is to communicate with the known signature's database to determine whether the signature is new or has been encountered already (either by the same MAVs or a different one). If the database determines the signature is new, the MAV does not perform the second step and does not compute weights (however, the signature is stored in the database and a new particle filter is created; see section 5.3.3 for details.) 2) On the other hand, if the database determines that a signature snapshot points is taking place, the second step starts. First, on the database side, the particle filter estimating the position of the signature being revisited is updated (see section 5.3.3 for details.) After the RF-signature particle filter has been updated, each particle in the MAV particle filter is assigned a weight as follows. A GMM is created starting from the particles in the signature being matched. Such a GMM is a bidimensional probability density function with the following equation:

$$f_{GMM}(x,y) = \frac{1}{M}\sum_{i=1}^{M} f_{\mathcal{N},\Sigma}^{i}(x,y) \tag{5.7}$$

where $f_{\mathcal{N},\Sigma}^{i}$ is a bidimensional Gaussian distribution with mean $\mu = [s_x^{[i]}\ s_y^{[i]}]^T$ and covariance matrix Σ (a diagonal matrix with value 2 on the main diagonal). Then, each particle is assigned the weight

$$w_t^{[i]} \leftarrow f_{GMM}(c_{xt}^{[i]}, c_{yt}^{[i]}). \tag{5.8}$$

After all weights have been computed, resampling can take place as described in Algorithm 2.

Adding Particles Using Coarse Map: Due to the unavoidable errors in the estimation process, I implemented an additional step to counter the formerly mentioned particle depletion problem. After the new set $\mathcal{X}_t$ has been created, I determine the location with the highest number of particles. Let v_d be this location, and let N be the set of neighbor nodes according to the coarse map. Then, the

25 particles with the lowest weight are discarded and replaced by an equal number of particles generated using a random distribution over the space associated with the nodes in N. The rationale behind this step is to generate particles to recover errors due to the erroneous determination that a transition from a room to the next effectively took place.

5.3.3 Particle Filter for Snapshot Points

I now describe how the spatial location of the signatures can be estimated using a set of particle filters. For the MAVs case, I do not compute a centralized estimation, but I rather associate a filter with each signature to be tracked. This estimation process has two main differences from the position and orientation estimation for the MAVs. First, the number of signature locations to be estimated is not known upfront. So new filters need to be created on-the-fly when a new signature to be localized is identified. Second, signatures do not move. Therefore the estimation process does not include a prediction step, only a correction step. As for the MAVs, each filter includes 100 particles.

Initialization from MAV Particles: As described in the previous subsection, a new signature is generated when the known signatures database receives a query from one of the MAVs with an RF-signature that cannot be matched to any of the formerly discovered ones. In this case, a new entry in the database is created and a new particles filter is instantiated. The initial set of particles for this new filter is copied from the particles of the vehicle that discovered the feature, while discarding the component related to heading because it is irrelevant for the signature estimation process.

Correction from MAV Particle Filter: Correction happens when a MAV queries the signature database with a signature that can be matched with one of the entries already discovered. In this case, Algorithm 2 is executed for the signature filter, with the exception of line 4, because no prediction takes place. The weight for $w_t^{[i]}$ for the ith particle is computed as follows. First, the position of the MAV that generated the snapshot point is determined by taking the average of its particles. Note that this average is implicitly weighted, because through the resampling process, particles with higher weight will be included more often in the particle set (see line 9 in Algorithm 2). As a result, they will be counted multiple times when computing

the average. Let $\overline{x}$ be the computed average position of the MAV generating the match, and let $s_{t-1}^{[i]}$ be the position of the ith particle in the signature particle filter at time $t-1$, and let $d_i = \left\| \bar{x} - s_{t-1}^{[i]} \right\|_2$ be the Euclidean distance between the expected position of the MAV and the particle. The weight of each particle at time t is then defined as

$$w_t^{[i]} = F_{d,\delta}(K) - F_{d,\delta}(-K) \tag{5.9}$$

where $F_{d,\delta}$ is the cumulative density function of a Gaussian distribution with mean d and variance δ. This formula is based on my experimental testbed showing that revisits are correctly detected when the displacement between the original and the new position is within K meters. The specific values for δ and K depend on the number of anchors and are further described in Section 5.4.5. Once weights have been computed, correction for the estimate of the signature particle filter can then take place through resampling, as described in Algorithm 2.

Database of Fingerprints: In order to help location estimation correction with snapshot points after each movement, the system maintains a database of fingerprints. A fingerprint at a specific location is a set of RSSI values from different stationary nodes measured by the MAV and stored in a dictionary data structure. When the node arrives at a new location, it calculates the cosine similarity between the newly discovered fingerprint and the fingerprints stored in the database. A pre-defined threshold T_{sig} is used to decide whether it is a new or known fingerprint.

5.3.4 DrunkWalk Planning

In this section, I describe how the system plans the paths of MAV nodes with location estimates of varying quality in order to deploy quickly. Figure 5.4 shows a flowchart of the planning algorithm.

Coarse Map: The system uses the layout with location of walls and doors of the environment to extract a coarse map. The doors are selected as the destination where I navigate the MAVs.

The coarse map makes very few assumptions about the quality of the map but provides a way to bias the motion of MAV nodes towards designated locations.

Entropy as Quality of Location Estimates: The entropy of a random variable x can be defined as the expected information that the value of x carries. In

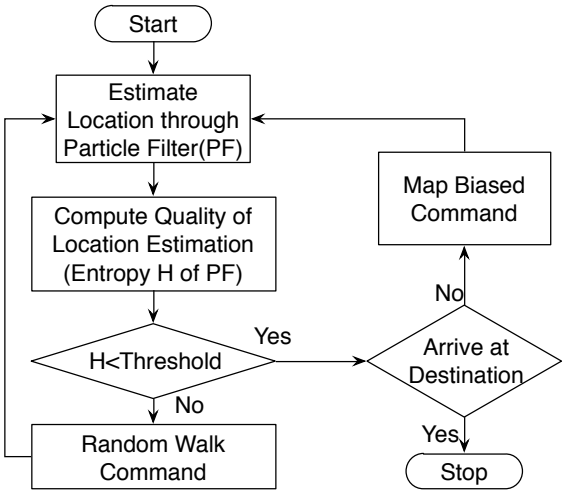


Figure 5.4: The figure shows the flowchart of the DrunkWalk planning algorithm. The planner adaptively changes between random walk and graph biased movement based on the entropy of particle filters tracking respective MAV nodes.

the discrete case, it is given by

$$H(x) = E[-\log_2 p(x)] \tag{5.10}$$

which represents the number of bits required to encode using an optimal encoding, assuming that $p(x)$ is the probability of observing x. The entropy can therefore be used as an indication of the uncertainty of the estimate of a particle filter. The lower the entropy the better the certainty of the location estimate is, and vice versa. For the particle filter, I calculate the entropy [100] of the weights at time t as

$$H_t = -\sum_{i=1}^{M} w_t^{[i]} \log_2 w_t^{[i]}. \tag{5.11}$$

Exploration: When the entropy of the particle filter is high ($>$ threshold T_H), the system seeks to primarily improve the location estimates. The intuition here is that with an incorrect estimation of current location, using the bias from the graph is likely to be incorrect. This also results in cases where the MAV may get stuck and can potentially perform worse than a purely random deployment strategy.

With this in mind, the planner employs a random walk strategy to direct the motion of MAV nodes. With random walk, the likelihood of nodes discovering snapshot points increases and so does the likelihood of improving their location

estimates. This is referred to as the exploration step.

Navigation: Correspondingly, when the uncertainty of location estimates is low ($H_t < T_H$), the planner commands the nodes to follow the bias indicated by the coarse map. With a more accurate location, likelihood of nodes following the bias and then reaching the intended destination increases.

A key point in choosing the directional bias from the graph is that it is sampled based on the distribution of particles in the node's particle filter. For example, consider a node with 20 particles indicating its position as room 1 and therefore requiring the node to go north-west to exit the room, while 80 particles indicate the node is in room 2 and must move south. In this case, the planner samples the movement direction according to the distribution of particles over the nodes of the graph, i.e., the node has a 20% chance of being commanded to move north-west and a 80% chance of receiving a south command.

Mode Configuration for Heterogeneous MAV Swarms: To adapt my algorithm to heterogeneous MAV swarms, where *advanced MAVs* and *basic MAVs* coexist, DrunkWalk also includes a heterogeneous mode configuration (HMC) module. The goal of HMC is to have *advanced MAVs* (with higher accuracy sensors) search the environment to improve the estimate and navigation accuracy for the whole team. To achieve that, DrunkWalk assigns a higher entropy threshold for *advanced MAVs*. As a result, *advanced MAVs* tend to enter the **Exploration** mode to help the whole team to search the environment. Once most parts of the environment are searched, the entropy threshold for *advanced MAVs* is set to the same value as *basic MAVs*. At this period, *advanced MAVs* are easy to navigate to the destinations.

Collision Recovery Strategy: MAV platforms have very limited sensing capability and often do not employ sophisticated obstacle detection sensors. Proposed MAV platforms [123] rely on their low weight and often use collisions themselves to discover obstacles. However, a strategy is needed in dealing with collisions so as to prevent MAV nodes from being stuck and enable them to back off from corners and crevices and seek out openings. This is especially useful when location estimates are inaccurate. The planner employs a random exponential back-off strategy, where nodes move in a randomly chosen direction (uniformly from a discrete number of directions) for a time duration that increases exponentially with the number of recent collisions. This is implemented by keeping a counter for collisions in a certain time-window. The counter is decremented with time if no new collisions are encountered.

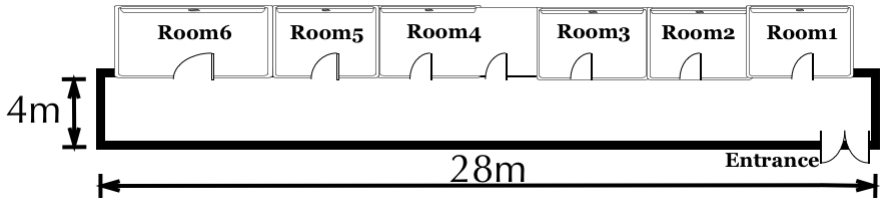


Figure 5.5: The figure shows the floor plan of 6 rooms with a hallway used for physical feature based simulation and real testbed experiments. The MAVs start from the entrance of the building and are navigated to different goal areas.

5.4 Evaluation

In this section, I evaluate the performance of my system in planning MAVs paths through physical feature based simulation and real experiments on a MAV testbed. Both simulation and real experiments are conducted in a building with multiple rooms connected with a hallway as shown in Figure 5.5. The MAVs start from the entrance and navigate to different goal areas (rooms). I first evaluate the system performance with both real test and physical feature based simulation on a homogeneous MAV swarm, where all the MAVs have the same sensing capabilities. Then, I test the influence of different system settings. Finally, I evaluate the performance of my system on a heterogeneous MAV swarm.

The evaluation focuses on the following aspects:

- Characterizing the performance of the system in terms of 1) navigation duration and 2) average accuracy of location estimations in comparison to existing navigation approach.
- Testing the influence of different settings on the system, such as number of stationary MAV nodes, noise of navigational sensors, radio fingerprint accuracy, and number of mobile MAV nodes.
- Validating the assumptions of the simulation experiments through real MAV testbed experiments.

For both testbed experiments and simulation, I compare my DrunkWalk algorithm to another online navigation strategy that does not require any location infrastructure, Dead-Reckoning with Map Bias (DRMB). Dead-reckoning with Map Bias is an infrastructure-free technique used to estimate a node's location in unknown environments [132]. This method uses measurements from motion sensors, optical

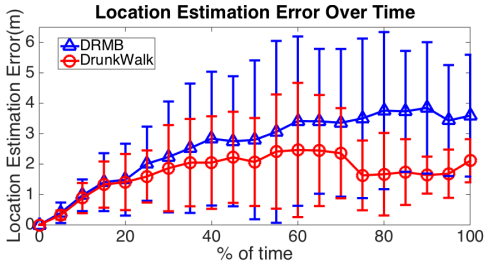


Figure 5.6: The figure shows the average and standard deviation of location estimation errors at different flying durations heading for the near destination using DrunkWalk and DRMB from 5 experiments. DrunkWalk achieves around 2m location estimation errors on average and 1-1.5m standard deviation.

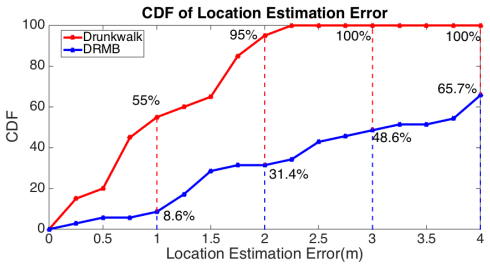


Figure 5.7: The figure shows the cumulative distribution function (CDF) of location estimation errors to arrive at the near destination using both DrunkWalk and DRMB from a typical run.

flow and gyroscope, to estimate the change in position of the node. Having an estimate of location, I then use the map to bias the direction of the node's movement similar to DrunkWalk.

5.4.1 Testbed Experiment Setup

To validate my system in a realistic setting, I implement my algorithm on a server and the SensorFly [123] [133] MAV testbed. The SensorFly platform used in my test has an 8-bit 16Mhz AVR AtMega128rfa1 micro-controller, a 3-axis accelerometer, a 3-axis gyroscope, a 2-axis optical flow velocity sensor, a 3-axis magnetometers, a ultrasonic altitude sensor and a XBee radio [134]. The platform has a flight time of 6-10 minutes. The SensorFly nodes are capable of translational and rotational motion directed by on-board PID control algorithms utilizing feedback from the on-board sensors.

In my setup, I manually fly 8 MAVs on a $4m \times 28m$ arena shown in Figure 5.5. Six nodes are allowed to disperse and deploy as beacons at initialization, while 2 nodes fly to seek out the destinations. The nodes are introduced to the rooms at the entrance and navigate to 3 kinds of goal destinations: near destination (room 2), medium destination (right door of room 4) and far destination (room 5). In addition, a laser range finder is used to track the location of the nodes as ground truth.

5.4.2 Testbed Experiment Results

I utilize the MAV testbed to illustrate the location estimation errors from short to long distances to compare performances and robustness of DrunkWalk with that of DRMB.

Figure 5.6 compares the performance and robustness of DrunkWalk and DRMB at different phases of navigation duration. I plot the average and standard deviation of location estimation errors from 5 experiments at different % of navigation duration. It is noted that I stop the experiments at 600 seconds even if the drones do not arrive at the destination since this is the typical flying time of a SensorFly node. At the first 20% of the navigation duration, DrunkWalk performs similarly as DRMB due to lack of snapshot points to correct location estimation errors. After this initial period, DrunkWalk's snapshot point correction maintained location estimation errors within the range of 1.5m to 2m. In contrast, the error of DRMB kept accumulating to larger than 3m. This is because multiple measurements at snapshot points can correct the location estimation errors in DrunkWalk. In addition, after 30% of the navigation time, the standard deviation of DrunkWalk location estimation is also $30\% - 60\%$ smaller than DRMB approach. This shows DrunkWalk is more reliable than DRMB.

Figure 5.7 shows cumulative distribution function (CDF) of location estimation errors using DrunkWalk and DRMB from a typical experiment. In DrunkWalk, corrections from snapshot points help keep errors within 2.5 meters. In comparison, location estimation error of DRMB remains unbounded. More than 50% of the time, DRMB has more than 3 meters error, compared to less than 1 meter error for DrunkWalk.

To illustrate the performance of DrunkWalk and DRMB for medium and far destinations, I use Figures 5.8 and 5.9 to show single run experiment results. I further evaluate the performance through simulated results in section 5.4.3. Note that except for the node using DRMB in the upper plot in Figure 5.9, the lines end when nodes reach their destination. The mobile node 1 failed to arrive at the far destination before 600 seconds when the battery was exhausted.

For both medium and far destinations, in the first 20 seconds, similar to navigation to near destination, DrunkWalk has similar location estimation errors with DRMB. After this initial period, adequate snapshot points help DrunkWalk to limit the

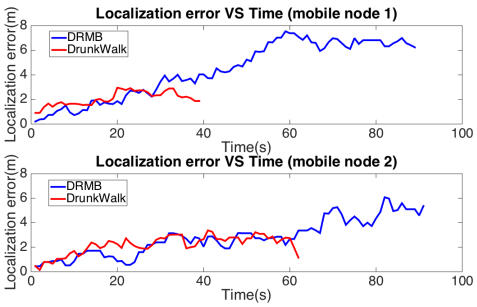


Figure 5.8: The figure shows the location estimation error over time using DrunkWalk and DRMB to the medium destinations. Mobile nodes with DrunkWalk arrive at the destination earlier than those with DRMB due to their capability to limit the location estimation errors.

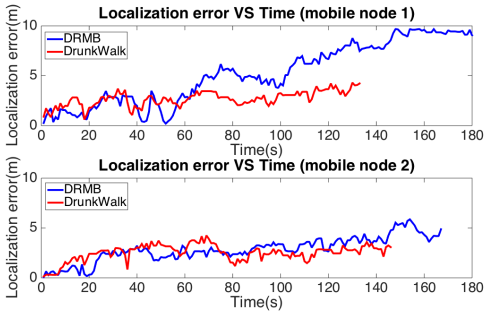


Figure 5.9: The figure shows the location estimation error over time using DrunkWalk and DRMB to the far destinations. It is noted that I do not plot all the data for mobile node 1 since it failed to arrive at the destination before the battery was exhausted (600 seconds).

location estimation errors while the error of DRMB keeps accumulating. This shows that multiple measurements at snapshot points in DrunkWalk do help limit location estimation errors. The higher location estimation accuracy from DrunkWalk leads to shorter (around 50%) navigation duration. It should be also noted that when using DRMB, mobile node 1 failed to arrive at the far destination before the battery died (600 seconds) while mobile node 2 arrived at the destination in around 170 seconds. This shows the unreliability of DRMB. On the other hand, both mobile nodes 1 and 2 with DrunkWalk arrived at the medium and far destination within similar durations. This shows the stability of DrunkWalk with the help of snapshot points.

5.4.3 Physical Feature Based Simulation Environment

I implemented a MAV simulation environment [135] for the SensorFly MAV indoor sensor swarm to evaluate my planning algorithms at large scale. The simulator includes a realistic physical arena, as well as sensor noise models, MAV mobility models and indoor radio signature collected from the testbed described earlier.

For my evaluations, I configure the simulator as follows:

- **Arena** – I use a multi-room indoor scenario shown in Figure 5.5, where nodes are required to autonomously navigate to different goal areas. I collect the radio fingerprints from the real arena and feed them into the simulation platform

to evaluate my system. This represents a typical indoor apartment scenario where such systems may be deployed in search and rescue applications. For more complex maps, I concatenate on portions of the map in figure 5.5.

- **Node Sensors** – The sensor nodes in the simulation are modeled after the SensorFly [123] MAV platform, which is also used in my testbed experiments described in section 5.4.1. Each node has a XBee 802.15.4 radio and Dead-Reckoning sensors – a gyroscope, an optical flow velocity sensor and an ultrasonic altitude measurement sensor. Noise models are obtained through empirical measurements on the testbed MAV platform.
- **Node Mobility** – The MAV nodes can turn by a commanded angle and move for a commanded time and velocity. I set the velocity to 1.0 m/s in accordance with the testbed MAV parameters. The velocity of course varies in accordance with the noisiness of the optical flow sensor that provides feedback to each MAV's control algorithm.
- **Simulation Time-steps** – The simulation time-step is chosen as $1sec$ that enables nodes to cover a distance of $0.8m$ to $1.2m$ in one simulation tick.
- **Radio** – The simulation supports estimating received signal strength (RSS) measurements between two nodes. The RSS is collected in the real scenario.
- **Destination** – I adopt the far destination (room 6) as the destination for simulation since this is the hardest situation which shows the baseline of the system performance and robustness.

All experiments were performed 25 times with 10 MAVs (6 stationary nodes and 4 mobile nodes) to evaluate both the performance and robustness of the system. I run the simulation for a time period of 600 seconds ($\tilde{1}$0 minutes) corresponding to the typical battery life of current generation MAV nodes.

5.4.4 System Performance

This section evaluates my system performance under different destination constraints and time limitations. A successful navigation is achieved when the node can be navigated to the destination within the given accuracy and time limitation. For example, if the destination coordinates in meters are (4, 5), the required accuracy is 1 meter and the time limit is 90 seconds, a successful navigation means that the

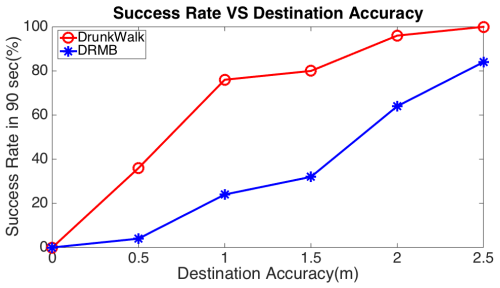


Figure 5.10: The figure shows the navigation success rate under different destination accuracy constraints within 90 seconds for DrunkWalk and DRMB.

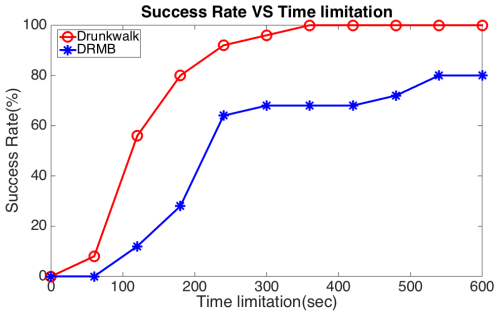


Figure 5.11: The figure shows the success rate as a function of time limitation under 0.5m destination accuracy constraints, using DrunkWalk and DRMB alone.

mobile node can arrive within a range of 1 meter from the destination (4, 5) within 90 seconds.

Figure 5.10 shows the navigation success rate within 90 seconds as a function of destination accuracy constraints using DrunkWalk and DRMB. When the destination accuracy is strict (0.5 meters), DrunkWalk achieves an acceptable success rate of around 40% while DRMB shows less than 5%. This means that, under very strict destination accuracy constraints, DRMB cannot achieve the required accuracy within the time limitation, while DrunkWalk can still work. For less constrained destination accuracies (1 to 2 meters), DrunkWalk shows consistently a 30% to 40% higher success rate than DRMB. Futhermore, DrunkWalk achieves 100% success rate for even looser destination accuracy constraints (2.5 meters) while DRMB achieves 84% success rate. This is expected since above 2.5 meter range is more than one half of the hallway width (4.0 meters), where even with high location estimation errors, the mobile node can still arrive at the destinations simply by following the walls.

Figure 5.11 plots the success rate as a function of time limitation for both DrunkWalk and DRMB under a 0.5m destination accuracy constraint. Under a strict time limitation (60 seconds), even DrunkWalk only achieved an 8% success rate since it was not able to get enough snapshot points to get accurate location estimation on the way to the destination. When the time limitations are extended (120 seconds to 300 seconds), DrunkWalk shows a 30% to 50% higher success rate than DRMB. This is because DrunkWalk has enough time to get snapshot points to correct the location estimation errors, while the error of DRMB keeps accumulating.

After 300 seconds, DrunkWalk achieves 100% success rate, while the success rate of DRMB becomes stable yet still below 80%. This means that even with loose time constraints, DrunkWalk still gets more than 20% higher success rate than DRMB.

5.4.5 Influence of System Settings

In this section, I evaluate how different system settings affect the location estimation errors under different system setups of both DrunkWalk and DRMB. I check the variance on 1) number of stationary MAV nodes, 2) navigational sensor noise, 3) radio fingerprint accuracy.

Number of Stationary MAV Nodes:

Figure 5.12 shows the location estimation errors for different numbers of stationary MAV nodes, where Drunkwalk achieves 1.5× to 6× reduction for average error and 1.5× to 4× reduction for standard deviation compared to DRMB. This shows its better capability and reliability to limit location estimation error. This can be attributed to improvement by matching snapshots with a larger number of stationary nodes broadcasting beacons. I can also see the decreasing trend for both average and standard deviation in Drunkwalk when the number of stationary nodes increases, which means that increasing the number of stationary nodes does help enhance performance.

Navigational Sensor Noise: The noise in motion measurements due to Dead-Reckoning with sensors is an important parameter in determining the eventual performance of the algorithm. Different MAV platforms and operating environments might have different amount of noise in their motion measurements, making it useful to analyze the performance of the algorithm for varying levels of sensor noise. For my simulations, in agreement with empirical measurements on my MAV platform, I model noise as a normal distribution with a standard deviation proportional to the sensor measurement [129], [131]. For the optical flow velocity sensor, a noise corresponding to a normal distribution with 0 mean and standard deviation of 20% of the measured velocity value was added. For the magnetometer, a 30° noise corresponds to a normal distribution with 0° mean and standard deviation of 30°. The resultant noise in DRMB location can be computed as per the motion update equation [129], [131]. I apply sensor noise to both DRMB and DrunkWalk estimates. Figure 5.13 and 5.14 show the location estimation error in DRMB and DrunkWalk

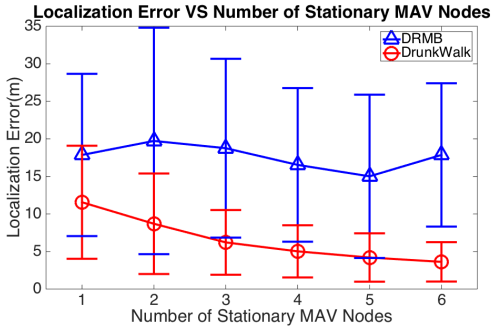


Figure 5.12: The figure shows the location estimation error with varying number of stationary MAV nodes using DrunkWalk and DRMB. DrunkWalk has an obvious decreasing trend when the number of stationary MAV nodes increases. It is noted that DRMB does not use stationary MAV and its performance variance in the figure is due to noise from sensors.

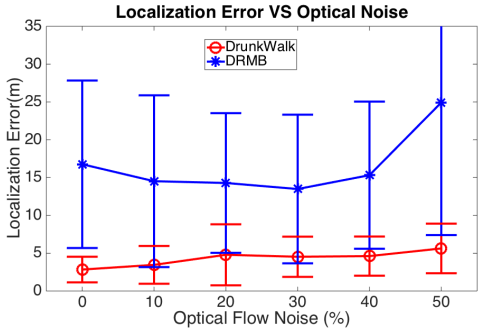


Figure 5.13: The figure shows location estimation error as a function of optical flow noise using DrunkWalk estimation and DRMB alone. The noise per sensor is modeled as a normal distribution with varying standard deviation. The plot shows that DrunkWalk is able to correct the DRMB error and maintain low standard deviation.

for 10 nodes at various sensor noise levels. In Figure 5.13, with optical flow noise level increasing from 0 to 50%, the average location estimation error of DrunkWalk increases from 2.5m - 5.5m while that of DRMB goes up very quickly to around 25 meters. The error increase is limited due to the corrections from the multiple measurements at the same snapshot points. In Figure 5.14, a faster increasing trend (from 0.5 to 7.5 meters) is observed with the magnetometer noise increase. In addition, similar increasing trends are observed for both DrunkWalk and DRMB. This indicates that high noise from the magnetometer weakens the correction capability on location estimation errors. However, during operation, raw magnetometer noise can be mitigated by gyroscope data.

Radio Fingerprint Accuracy: Location estimation error depends on the resolution to identify snapshots on the node paths. This is accomplished by using radio signatures from stationary MAV nodes deployed at the beginning. When a radio signature collected by a node at a certain location is similar to a fingerprint in the database: the system classifies it as a snapshot and performs the correction of location estimates. Thus, the performance of DrunkWalk depends on the resolution of the fingerprint matching, i.e., the area or distance within which two radio fingerprints can be reliably classified as being at the same location. If the criteria is too loose,

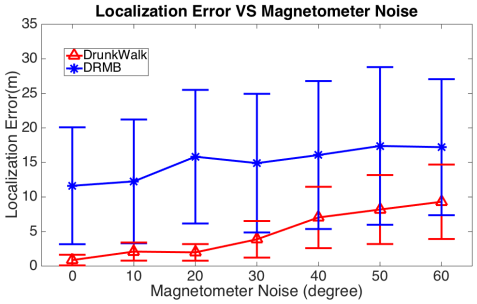


Figure 5.14: The figure shows location estimation error as a function of magnetometer noise using DrunkWalk estimation and dead reckoning alone. The noise per sensor is modeled as a normal distribution with varying standard deviation. The plot shows that DrunkWalk is able to correct the DRMB error and maintain low standard deviation.

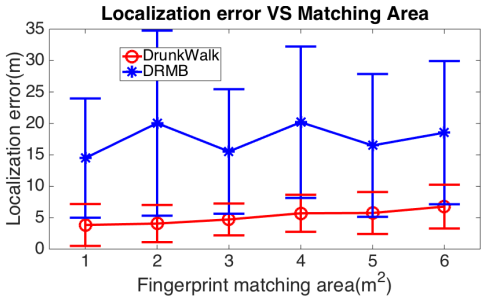


Figure 5.15: The figure shows location estimation error as a function of signature matching area for DrunkWalk and DRMB. To be noticed, DRMB does not work with stationary MAV and noise from sensors cause the performance variance.

incorrect points may be matched together, which leads to few snapshot points.

Figure 5.15 shows location estimation error as a function of signature matching area for DrunkWalk and DRMB for 10 nodes. I observe that DrunkWalk offers an improvement of over 3× compared to DRMB even for a poor matching resolution of $6m^2$. In case of MAV navigation, where the low-cost of nodes makes it possible to deploy a relatively large number of beacon nodes and attain high fingerprint accuracies of around $1m^2$, DrunkWalk provides a much larger reduction in error.

5.4.6 Heterogeneous System Performance

This subsection evaluates how my DrunkWalk works with heterogeneous MAV swarms. For simplicity but without loss of generality, I focus on swarms consisting of two different MAVs: *advanced MAVs* (with more accurate sensors) and *basic MAVs* (with less accurate sensors). In the default setting, I have 10 mobile MAVs in the swarm, 3 of which are *advanced MAVs*. I set the sensing noise of *advanced MAVs* at 20% for optical flow noise and 20° for magnetometer noise. I set the sensing noise of *basic MAVs* twice as that of *advanced MAVs*, i.e. 40% optical flow noise and 40° magnetometer noise. I check the system performance under variant 1) percentage of *basic MAVs* in the swarm 2) noise level ratio between *advanced MAVs* and *basic MAVs* 3) total number of mobile MAVs nodes.

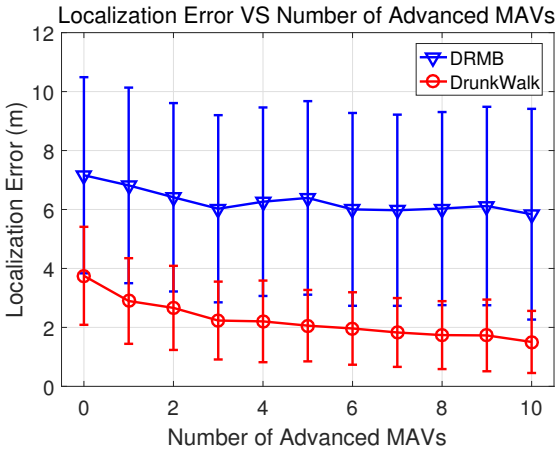


Figure 5.16: The figure shows the localization estimation error of MAV swarms consisting of different numbers of *Advanced MAVs*. The total MAV number in the swarm is fixed as 10.

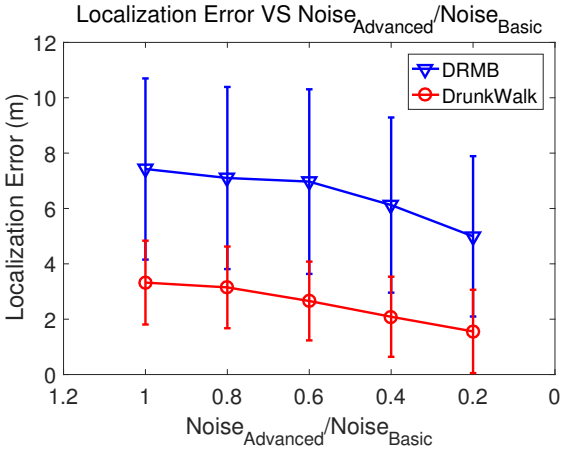


Figure 5.17: The figure shows the localization estimation error as a function of noise ratio between *Advanced MAVs* sensors and *Basic MAVs* sensors, using DrunkWalk and DRMB.

Number of *Advanced MAVs*: To check the influence of the number of *Advanced MAVs* on system performance, I plot the location estimation error in Figure 5.16. Since I fix the total number of mobile MAVs as 10, when the number of *advanced MAVs* increases, the number of *basic MAVs* decreases. First, the errors of both DrunkWalk and DRMB present descending trends when the number of *advanced MAVs* increases. The error reduction of DrunkWalk comes from both sensing accuracy improvement and collaboration, while the error reduction of DRMB only comes from sensing accuracy improvement. Second, the error of DrunkWalk decreases faster than that of DRMB. Specifically, from 0 *advanced MAVs* to 10 *advanced MAVs*, the error of DrunkWalk reduces 60% (from 3.75m to 1.5m), while the error of DRMB reduces less than 20% (from 7.2m to 5.8m). This illustrates that DrunkWalk can take advantage of heterogeneous MAV swarms to reduce location estimation error. Finally, from 0 *advanced MAVs* to 10 *advanced MAVs*, the standard deviation of DrunkWalk errors decreases more than 40% (1.7m to 1m), while DRMB does not show too much difference on standard deviation of errors. This shows the reliability of my DrunkWalk with heterogeneous MAV swarms. Both the error reduction and reliability improvement come from DrunkWalk's assigning different roles to *advanced MAVs* and *basic MAVs* through different system configurations, which causes better collaboration for reducing and limiting location estimation error.

Noise Level Ratio Between *Advanced MAVs* and *Basis MAVs*: To further investigate the heterogeneous system, I examine the location estimation error when the noise of *advanced MAVs* varies. In this experiment, the noise of *basic MAVs* sensors is fixed at default settings. Figure 5.17 shows the estimated error with varying noise of *advanced MAVs* sensors. The x-axis represents the ratio of two different kinds of noise. The errors of both DrunkWalk and DRBM decrease when the sensors of *advanced MAVs* become more accurate. Specifically, from 1 to 0.2, DrunkWalk error decreases more than 50% (from 3.3m to 1.5m), while DRMB error decreases 33% (from 7.5m to 5m). This is because that DrunkWalk assigns different roles to *advanced MAVs* and *basic MAVs* through different system configurations, which causes better collaboration to improve location estimation accuracy.

Total Number of Mobile MAV Nodes: Figure 5.18 presents the location estimation error with a varying total number of MAVs in a swarm. The noise levels for *advanced MAVs* and *basic MAVs* are set as default and the portion of *advanced MAVs* in the swarm is set as 30%. I increase the total number of mobile MAVs from 10 to 30. With the total number of MAVs increased, error of DRMB does not show too much difference, while error of DrunkWalk decreases 40% (from 2.3m to 1.4m). In addition, the error standard deviation of DrunkWalk decreases 15% (from 1.36m to 1.17m) and the error standard deviation of DRMB still does not have obvious change. This is because DRMB does not adopt collaboration between MAVs. As a result, increasing the total number of the MAV swarm does not help reduce estimation error or improve stability. In contrast, my DrunkWalk not only adopts collaboration between MAVs, but also assigns different roles to *advanced MAVs* and *basic MAVs* through different system configurations, which helps to improve the system performance on accuracy and reliability.

5.4.7 Related Work

Works related to DrunkWalk mainly fall into three domains: sensor networks, robotics and mobile computing.

The sensor network domain has a number of works on deploying and navigating mobile sensors [136, 137, 138, 139]. Howard et al. [140, 141] present techniques for mobile sensor network deployment in an unknown environment. Their approach constructs fields such that each node is repelled by both obstacles and by other

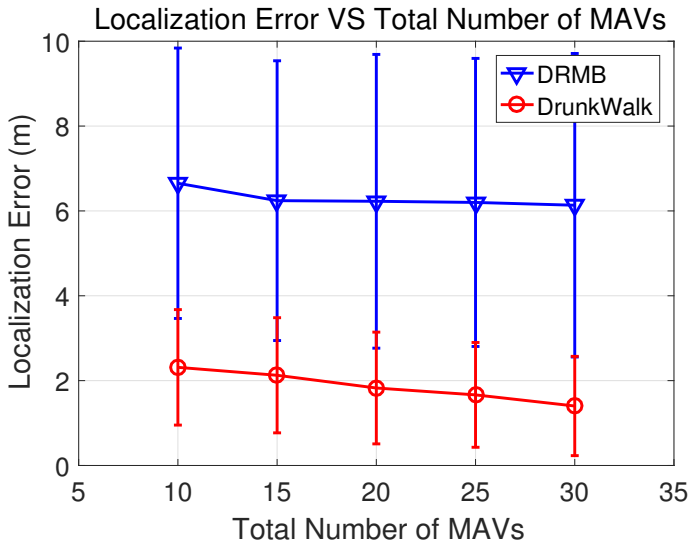


Figure 5.18: The figure shows the localization estimation error as a function of total number of MAVs in the swarm. The noise levels for *advanced MAVs* and *baisc MAVs* are set as default and the portion of *advanced MAVs* in the swarm is set as 30%.

nodes, enabling the network to spread itself throughout the environment. Similarly, Batalin et al. [142] present a deployment algorithm for robot teams without access to maps or location. The robots are assumed to be equipped with vision sensors and range finders and select a direction away from all their immediate sensed neighbors and move in that direction. The algorithm does not require communication between nodes but also does not allow nodes to be deployed at designated locations. The domain experts have no control over the emergent deployment locations of the nodes.

The problem addressed in this section can also be seen as an instance of the Simultaneous Localization And Mapping (SLAM) problem that has been extensively studied in robotics [100, 143, 144, 145]. In fact, in the system I described multiple MAVs try to localize themselves while at the same time trying to acquire a representation of the spatial distribution of the radio signatures. In recent years there have been copious research in SLAM using either methods based on Kalman filters [146, 147, 148, 149] or particle filters [150, 151, 152, 153]. Both approaches, however, have been mostly applied to solve instance of the SLAM problem where mobile agents are equipped with sensors returning distances (e.g., laser range finders, or sonars) or cameras (either monocular or stereo). Therefore, the ultimate objective of these solutions to the SLAM problem was to map physical entities located in the environment, like walls, obstacles, etc. Methods based on Kalman filters are

not applicable for the scenario I consider because I am dealing with multimodal, nonparametric probability distributions. Therefore, I opt for a solution based on particle filters.

Approaches based on explicit perception and processing of radio signals have been mostly aimed at implementing localization systems with the underlying assumption that radio signals were preliminarily collected off-line to build so-called *map signals* [154, 155, 156, 157]. A recent paper by Twigg et al. [158] discusses a system where a robot autonomously discovers the area within which connectivity with an assigned WiFi base station is ensured. Their solution, however, solves only the mapping side of the problem because the robot is equipped with a laser range finder solving the localization problem. In other words, RSS readings are mapped to the physical space exploiting the availability of a different sensor providing reliable localization.

For people carrying mobile devices, SLAM-like approaches have recently been proposed that fuse WiFi-based RSS and motion sensor data to simultaneously build a sensor map of the environment and locate the user within this map. e.g. radio fingerprint maps [159, 160, 161, 162], or organic landmark maps [163, 164]. These approaches focus on the location estimation part of an orthogonal problem, where the motion of users cannot be controlled and hence, does not involve motion planning or deployment. Purohit et al. [129] present a system for infrastructure-free single room *sweep* coverage with MAV sensor swarms. Their approach, however, does not involve the concept of location estimation and navigation and does not support navigating nodes to pre-assigned destinations.

To the best of my knowledge, this section presents the first attempt to solve a SLAM problem using a swarm of MAVs that combines location estimation and adaptive planning to improve the success rate and accuracy of navigation.

5.5 Conclusion

This chapter presents a system for collaborative and adaptive planning of resource-constrained MAV sensing swarms to quickly and efficiently navigate to preassigned locations. The system utilizes collaboration between nodes of the swarm to overcome the sensing and computational limitations of MAV nodes, and the challenging operating environments. In addition, to extend my algorithm to heterogeneous drone

swarms, my algorithm assigns different roles to *advanced MAVs* and *basic MAVs* through different settings based on their sensing capabilities. I comprehensively evaluate the system through large-scale simulations and real MAV testbed experiments, showing that DrunkWalk achieves up to 6× reduction in location estimation errors, and as much as 3× improvement in navigation success rate under the given time and accuracy constraints. All these results validate the effectiveness of applying spatiotemporal relationships on controllable large scale mobile CPS for estimating system status and route planning.

Chapter 6

Actuation Planning for Semi-Controllable Large Scale Mobile CPS

This chapter introduces how to utilize spatiotemporal relationships to address the challenge of route planning for semi-controllable systemsto achieve high sensing coverage quality on the vehicle based CPS, as described in Chapter 1. Compared to the controllable large scale mobile CPS in Chapter 5, route planning is more challenging for semi-controllable systems, since their route planning involves estimation of both system status and physical field. Along with the challenges of controllable large scale mobile CPSs, semi-controllable systems have to face the problem of selecting which mobile devices should be actuated in route planning and how to design routes for the selected mobile devices.

The system predicts the vehicles' mobility to help the system to choose the "correct" vehicles to actuate (ST-PHY), and the ride requests over the city to help lower the incentive cost (ST-SYS). The system combines information of mobility of all the vehicles and the ride requests over the city to calculate an optimal solution for route planning (ST-PHY-SYS).

6.1 Problem Overview

The rapid growth of mobile devices with powerful sensing units has promoted the development of mobile crowdsensing (MCS), in which spatially distributed participants collectively sense and share data. The extracted information from shared data can be used to measure, map, analyze, or estimate any processes of interest, such as traffic conditions, air pollution, noise level, etc [165, 166]. MCS brings many advantages to sensing systems, including low deployment cost, accessible large scale, and easy maintenance [167].

Vehicle fleets are an important platform for MCS due to their high mobility and large range. Especially, traditional taxis and new forms of taxis (Uber, Lyft and Didi) operate throughout the city with long operational time. These fleets enable large scale sensing data with high spatiotemporal coverage and make a lot of urban sensing applications feasible [168].

Sensing coverage quality, which considers both amount and distribution of data collection, is one of the key performance indices (KPI) of the MCS system that influences the quality of the information collection [169]. Good quality data collection requires both large and balanced coverage in the spatial and temporal domain [21]. Large coverage ensures sufficient information is collected, while balanced coverage ensures informative data collection.

As non-dedicated sensing platforms, MCS systems using taxis do not guarantee good sensing coverage quality even with large numbers of taxis. This is because most taxis gather around busy areas, like central business districts (CBDs), while little data are collected in other areas [17]. A lot of past work has been done to improve the sensing coverage quality. Auction-based or game-theoretical mechanisms have been proposed to actuate MCS participants [170, 171, 172, 173]. These approaches require the participants to select and bid on the task. These approaches rely on a large number of rational participants and incorporate all their preferences. As a result, they are particularly sensitive to driver participation and attention. Furthermore, they do not incorporate the future mobility of the unselected vehicles effect on the overall sensing coverage quality, which brings a lot of uncertainty to the effectiveness of the sensing coverage quality after actuation.

It is difficult to optimize sensing coverage quality in a vehicular MCS with a limited budget due to **two major challenges: high uncertainty on actuation**

effectiveness, and conflicting goals between the vehicle fleet and MCS platform.

- *High uncertainty on actuation effectiveness:* With limited budget, only a small percentage of the whole vehicle fleet can be actuated and the future mobility of the rest of the vehicles is not considered. As a result, the effectiveness of actuation is highly uncertain.
- *Conflicting goals:* As a non-dedicated sensing platform, taxis make individual optimal decisions on looking for new ride requests (customers), which makes them gather in the busy areas with more ride requests. This leads to much fewer taxis showing up and less data being collected in other parts of the city. As a result, simply actuating taxis without a monetary incentive causes high actuation cost and low motivation [174].

This section answers the question: how can one efficiently actuate non-dedicated sensing platforms (ride-based vehicles) to achieve optimal sensing coverage quality with a limited budget? I present ASC, a system that actuates vehicular taxi fleets for optimal sensing coverage quality by matching ride requests with taxis. ASC determines routes for all the available taxis through two main steps. 1) The system first adopts a mobility prediction model to forecast the near-future taxi destinations. The prediction guides the system to decide which taxis to select for actuation to achieve maximum sensing coverage quality improvement. The system intends to spend its budget on taxis which are predicted to head for busy areas (instead of those heading for sparse areas), and actuates them to sparse areas. As a result, actuating one taxi brings more sensing quality improvement. 2) ASC includes a ride request prediction model to predict near-future ride requests across the city. Based on this prediction, the system chooses routes to actuate taxis, which aims to improve the overall sensing coverage quality and match the ride requests with the taxis. This not only lowers the cost of actuation but also improves the motivation for the driver [174]. Utilizing these two key steps, the system sends the actuated routes and corresponding monetary incentives to the selected taxis.

6.2 Problem Definition

In this section, I discuss the problem of optimizing sensing coverage quality in vehicular MCS. My actuation system is not dependent on the particular application and can be used for any type of high-level vehicular MCS tasks. First some preliminary definitions are given. Then I describe the goal of my system. Finally, I formulate the problem of optimizing sensing coverage quality.

According to spatial and temporal resolution setup (d_s and d_t), the system discretizes the focus rectangle area into n_x by n_y congruent *grids* (x_i, y_j) and *time slices* t_k. The longitude, latitude and time index are represented by x_i, y_j and t_k respectively. I note that according to the average taxi speed, d_s and d_t are set so that a taxi covers at most d_s within d_t.

6.2.1 Key Definitions

Worker: Denoting a taxi's fleet as C, each worker $c \in C$ represents a taxi carrying sensors for different applications. The worker drives within the map of the target city and continues to collect data during its trajectory. The spatial coordinate of c at time t is denoted as (x_t^c, y_t^c) and obtained by global positioning system(GPS).

Actuation period: The actuation period $T = nd_t$ denotes the time length for the selected taxis to finish the actuation route. For simplicity but without loss of generality, I set $T = 5d_t$ in this paper. The changing of n does not change the problem and solution.

Actuation Task: An actuation task for a worker c refers to a route that a worker is asked to cover within an actuation period T. The route consists of a sequence of coordinates for each d_t during the actuation period and expressed as

$$\left\{(x^c{}_\tau, y^c{}_\tau), (x^c{}_{\tau+d_t}, y^c{}_{\tau+d_t}), ..., (x^c{}_{\tau+T}, y^c{}_{\tau+T})\right\},$$

where $(x^c{}_\tau, y^c{}_\tau)$ is the original location of the taxi c when the actuation task starts at time τ.

Actuation Availability: At the beginning of each actuation period T, each worker c reports its actuation availability. An available worker means there is no passenger in the taxi and the driver is willing to follow the assigned trajectory with the given monetary incentive. A worker is called an "actuated worker" when it accepts an

actuation task, and a "non-actuated worker" otherwise.

Budget & Monetary Incentive: The budget R is the total amount of money available to actuate workers during each actuation period. When a worker c is assigned an actuation task, a monetary incentive $B(c)$ is also allocated. The total monetary incentives do not exceed the given budget R.

Sensing Coverage: The sensing coverage A refers to the set of data points collected by all workers during one actuation period T, including both "actuated" and "non-actuated" workers.

6.2.2 Actuation Objective

The objective of actuation is to achieve optimal sensing coverage quality (a large amount of well-balanced data) by actuating part of the vehicle fleets with given limited budget. I define the sensing coverage quality $\phi(A)$ as a combination of the total amount of sensed data and the "balance level:" the sensed data distribution over the covered area. The balance level represents how uniformly the sensed data are distributed in both the temporal and spatial domains. I quantify this using the entropy of the sensed data distribution. Thus, the overall sensing coverage quality is obtained by the weighted sum of the total amount of sensed data and the entropy of their distribution.

Eq. 6.1 gives the mathematical formulation of $\phi(A)$, where $E(A)$ is the entropy of data distribution (data balance), and $Q(A)$ is the number of data points. $\alpha \in (0, 1)$ is tuned to be large when balancing the data distribution is the main focus of the actuation task, and small when the main focus is collecting a large amount of data.

$$\phi(A) = \alpha E(A) + (1 - \alpha)\log Q(A) \tag{6.1}$$

6.2.3 Problem Formulation

To optimize the sensing coverage quality $\phi(A)$ with limited budget R, the system needs to 1) select the "correct" taxis to actuate that efficiently utilize the budget and 2) plan the actuation task routes for each selected taxi. Therefore, I give the mathematical formulation of the actuation problem at time t as:

$$\max_{I(c_i),\left\{(x_c^t,y_c^t),\ldots,(x_c^{t+T},y_c^{t+T})\right\}} \phi(A) \tag{6.2}$$

$$s.t.\ 0 \leq x_{c_i}^{\tau} \leq n_x d_s, \mathrm{t} \leq \tau \leq t+T, \mathrm{i} = 1, \ldots |C| \tag{6.3}$$

$$0 \leq y_{c_i}^{\tau} \leq n_y d_s, \mathrm{t} \leq \tau \leq t+T, \mathrm{i} = 1, \ldots |C| \tag{6.4}$$

$$|x_{c_i}^{\tau} - x_{c_i}^{\tau-d_t}| \leq d_s, \mathrm{t} \leq \tau \leq t+T, \mathrm{i} = 1, \ldots |C| \tag{6.5}$$

$$|y_{c_i}^{\tau} - y_{c_i}^{\tau-d_t}| \leq d_s, \mathrm{t} \leq \tau \leq t+T, \mathrm{i} = 1, \ldots |C| \tag{6.6}$$

$$\sum_{i=1}^{|C|} B(c_i) \cdot I(c_i) \leq R \tag{6.7}$$

$I(c_i) = 1$ represents that worker c_i is selected for actuation, and $I(c_i) = 0$ that worker c_i is not selected. Eq. (3) and (4) constrain the system to only consider workers' mobility within the focus area. Eq. (5) and (6) constrain that each worker cover at most d_s within d_t. Eq. (7) constrains that total monetary incentives must not exceed the given budget R.

In my system, at the beginning of each actuation period T, taxis automatically report their information including: taxi id, current location, and actuation availability for the coming actuation period. Based on the reported information, tasks and monetary incentives are calculated and assigned to selected available taxis. It is assumed that taxis follow the actuation task routes until the end of the the actuation period if they accept the monetary incentive.

6.3 System Design

This section introduces how I utilize spatiotemporal relationships to design the actuation system to optimize sensing coverage quality. I first discuss how I integrate the mobility prediction model (ST-SYS) and the ride request prediction model (ST-PHY) into the system in Section 6.3.1. Then, I discuss how to design vehicles' mobility prediction and ride request prediction in detail in Sections 6.3.2 and 6.3.3. Finally, I discuss the design of the monetary incentives and multi-incentive algorithm (ST-PHY-SYS) in Sections 6.3.4 and 6.3.5.

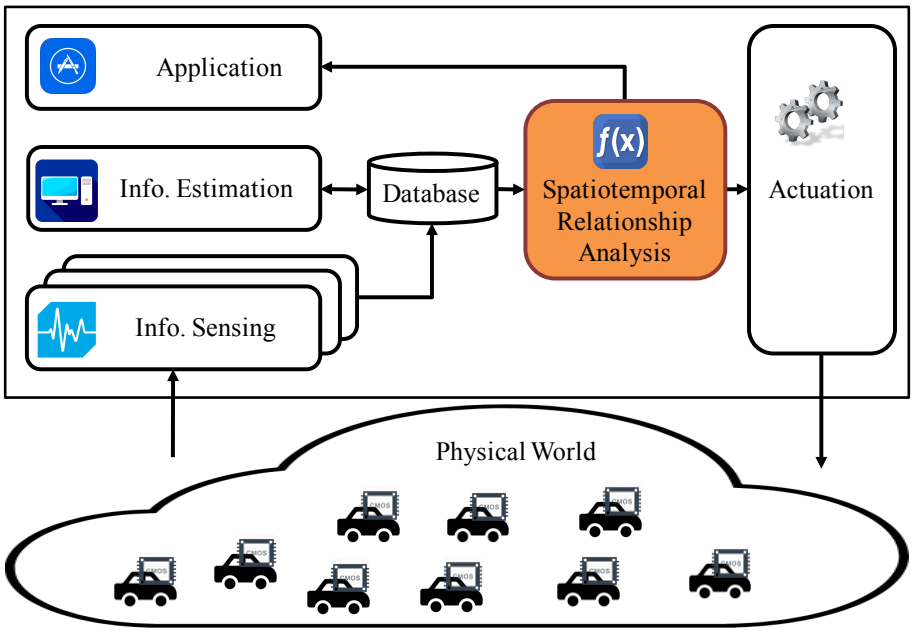


Figure 6.1: The figure shows the architecture of a spatiotemporal relationship-aided large scale mobile CPS implemented as a semi-controllable vehicle based CPS for route planning (improving sensing coverage quality in MCS), which involves estimation on system status and physical field.

6.3.1 System Overview

Figure 6.1 shows the high-level architecture of the semi-controllable vehicle based CPS. The *information sensing layer* gets the sensing data, taxi status and trajectory of vehicles and stores them in the *database*. The taxi status indicates whether the taxi is available for actuation to ASC. Unavailability can occur for two reasons: customers already riding in taxis, or drivers' unwillingness to be actuated. The *information estimation layer* processes the sensing data in the *database* and stores the results back to the *database* for the *application layer*. The *spatiotemporal relationship analysis layer* gets the status and trajectory data of taxis from the *database* and calculates an optimal actuation solution for the *device actuation layer* for improving sensing coverage quality.

To optimize sensing coverage quality, I design the actuation system based on two key observations: **1) The cost of actuating one taxi depends on whether the system can match a taxi with a ride request at the destination.** If the system matches the taxi with a ride request, the taxi driver is willing to accept a

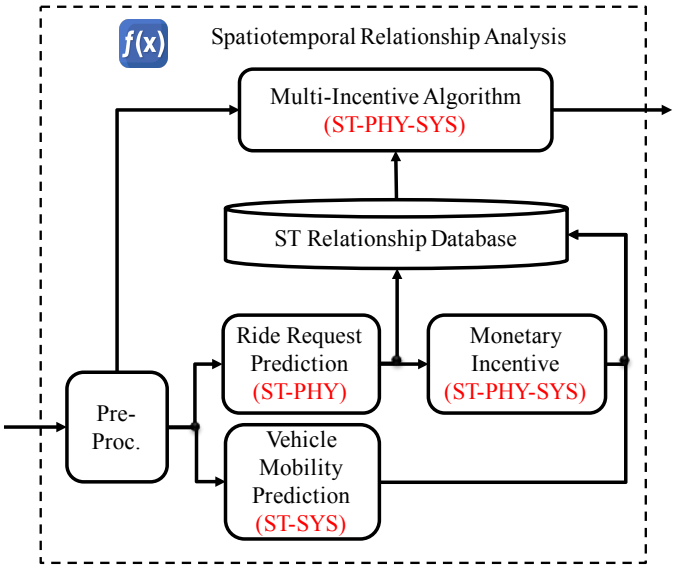


Figure 6.2: This figure shows the architecture of applying the *spatiotemporal relationship analysis layer* on semi-controllable large scale mobile CPS (vehicle based CPS) for route planning (improving sensing coverage quality in MCS), which involves estimation of system status and physical field. The *spatiotemporal relationship analysis layer* predicts the vehicles' mobility to help the system choose the "correct" vehicles to actuate (ST-PHY), and predicts the ride requests over the city to help lower the incentive cost (ST-SYS). The system combines information about mobility of the vehicles and the ride requests over the city to calculate an optimal solution for route planning (ST-PHY-SYS).

lower monetary incentive since they can earn money from the new rides [174]. **2) The sensing coverage quality after actuation depends on selecting which taxis to actuate.** The system does not have to actuate taxis that plan to head for sparse areas, as changing their trajectories would not significantly improve the sensing coverage quality. On the other hand, changing the trajectories of those that plan to head for busy areas and actuating them towards sparse areas improves sensing coverage quality more.

Therefore, I integrate two prediction models into the system. The mobility prediction model forecasts the mobility of taxis and offers guidance for the system to wisely select which taxis to actuate. To be specific, the system selects taxis heading for dense areas and actuates them to sparse areas, which leads to a higher sensing coverage quality improvement. The ride prediction model forecasts the coming ride requests over the city. When taxis are matched with ride requests, taxi drivers are willing to accept a lower incentive. As a result, the cost of actuating a taxi is lowered

and more taxis can be actuated for better sensing coverage quality with the same budget.

Figure 6.2 shows how the system integrates the two prediction models for actuation. Taxis report their real-time trajectory data and whether they are available for actuation to ASC. Unavailability can occur for two reasons: customers already riding in taxis, or drivers' unwillingness to be actuated. ASC calculates 1) which taxis to be actuated, 2) where they will be actuated to, 3) how much monetary incentive they are paid, and potential ride requests at the actuation destination. The results are sent back to the taxis, thus actuating them to achieve sensing coverage quality optimization.

The *Pre-Processing* module discretizes the focus rectangle area of the city and the time with the given spatial and temporal resolution (d_s and d_t). According to the average taxi speed, d_s and d_t are set so that a taxi covers at most d_s within d_t.

The *Vehicle Mobility Prediction* module, which is trained by each taxi's history trajectory data, predicts taxi mobility. The prediction output is fed to the *Multi-Incentive Algorithm* module to guide the system to wisely select the taxis to actuate, which improves the effectiveness of the actuation. **ASC allows different mobility prediction models. For simplicity but without loss of generality, this paper adopts a Markov based mobility prediction model.** The details can be found in Section 6.3.2.

The *Ride Request Prediction* module predicts ride requests over the city, whose results are sent to the *Multi-Incentive Algorithm* module. Based on this prediction, the *Multi-Incentive Algorithm* module selects routes for actuated taxis. The *Ride Request Prediction* module uses historical ride request data, which can be derived from taxi occupancy data, to train the ride request model. The system framework allows for different ride request prediction models. For simplicity but without loss of generality, this paper adopts a graph-based ride request prediction model, whose details will be discussed in Section 6.3.3.

The *Monetary Incentive Calculation* module calculates the incentive based on the selected routes from the *Multi-Incentive Algorithm* module and the prediction from the *Ride Request Prediction* module. The results are sent back to the *Multi-Incentive Algorithm* module for further optimization. The details can be found in Section 6.3.4.

The *Multi-Incentive Algorithm* module selects the taxis to be actuated and designs trajectories for those taxis by collaboratively considering 1) taxi mobility predictions

from the *Vehicle Mobility Prediction* module, 2) ride request predictions from the *Ride Request Prediction* module and 3) monetary incentive from the *Monetary Reward Calculation* module. The details will be discussed in Section 6.3.5.

6.3.2 Mobility Prediction Model

The mobility prediction model offers information used to guide the system to select taxis for actuation. As I have introduced in the previous section, the trajectories I build for each taxi are discrete in both the spatial and temporal domains. Therefore, I adopt a Markov Chain (MC) model, which is widely used for modeling transitions within discrete states [175]. In an MC model, each taxi corresponds to a transition kernel P that describes its mobility pattern. The entry P_{ij} represents the probability that the taxi moves from location i to location j. Each row P_i of the transition kernel represents the probability distribution of the taxi moving from location i to its next location.

As for the training of an MC model, given a trajectory of the taxi, the maximum posterior estimation of P_{ij} is as follows:

$$\hat{P}_{ij} = \frac{n_{ij} + \beta}{\sum_{j'} n_{ij'} + \beta}, \tag{6.8}$$

where n_{ij} represents the number of times the taxi moves from location i to location j, and β is a smoothing coefficient to avoid dividing by zero.

Once the estimated transition kernel $\hat{P}$ is acquired, it is used to predict the taxi's future movements. The process can be formulated as: given a taxi's current location $\overrightarrow{x_0}$, the estimated transition kernel $\hat{P}$, and a possible trajectory $\overrightarrow{x}_{(1:n)}$, the probability $p(l)$ that the taxi moves along this trajectory in the future is calculated as:

$$p(l) = \prod_{i=1,2,..,n} \hat{P}_{\overrightarrow{x}_{i-1}\overrightarrow{x}_i} \tag{6.9}$$

However, the size of the transition kernel correlates with the number of distinct locations visited by each taxi, which means the computation complexity drastically increases. Also, I discovered that most transitions happen within two connected grids, due to the speed limit in the city. Therefore, in order to reduce the computational

Algorithm 3: Training Algorithm for Vehicular Mobility Prediction

Input: Trajectory $X = x_1, x_2, ...x_n$, grid-to-area Map $f(\cdot)$.
Output: Direction Transition Kernel D.
Initialize:
    Transfer X into transition directions $d = \{d_1, d_2, ..., d_{n-1}\}$
    Set each element of $n \leftarrow \alpha$
for $i \in \{1, ..., n-1\}$ **do**
    Map grid ID into *Area* ID $a_i \leftarrow f(x_i)$
    $n_{a_i d_i} \leftarrow n_{a_i d_i} + 1$
end
Calculate D using n based on (6.8)

work, I assume that each taxi either moves to an adjacent grid or stays in its current grid. This assumption decreases the length of each row P_i of the transition kernel to nine, representing the nine possible directions that a taxi can move to. I denote the new transition kernel as the direction transition kernel D.

Moreover, I noticed that taxis within adjacent grids tend to follow similar mobility patterns [176]. Thus, to further limit the computation complexity while keeping the same spatial resolution, I first partitioned the city into several non-overlapping sub-areas that are larger than a grid, and then let grids within the same area share the same transition probability distributions, i.e. the same row of D.

The mobilitiy prediction module is based on this mobility model and has two parts, namely training and predicting. In the training part, as shown in Alg. 3, a function $f(\cdot)$ maps the grid ID into the $sub-area$ ID. Then the training part takes a taxi's historical trajectory $X = x_1, x_2, ...x_n$ and the grid-to-sub-area map $f(\cdot)$ as inputs, and generates the direction transition kernel D as output. For each location of the trajectory, it maps the grid ID to the $sub-area$ ID, and then counts the transition direction in a count matrix n. Based on n, it uses Eq. 6.8 to estimate the direction transition matrix D. In the prediction part, as shown in Alg. 4, the probability that a taxi travels along a fixed trajectory l given its transition kernel D and current location x_0 is calculated. This probability is calculated as a product of the probability of each single transition in the fixed trajectory.

Algorithm 4: Vehicular Mobility Prediction Algorithm

Input: Direction Transition Kernel D, Current location x_0, Fixed trajectory $l = \{x_1, ..., x_m\}$,grid-to-area Map $f(\cdot)$
Output: Probability p.
Initialize:
    Transfer l into transition directions $d = \{d_0, d_1, ..., d_{m-1}\}$ (including current location x_0)
    $p = 1$
for $i \in \{0, ..., m-1\}$ **do**
    Map grid ID into area ID $a_i \leftarrow f(x_i)$
    $p \leftarrow p * D_{a_i d_i}$
end
Return p

6.3.3 Ride Request Prediction Model

My system requires a model to predict ride request numbers over locations and time in a city. This prediction enables the system to match ride requests with taxis, which makes taxi drivers willing to accept lower incentives. As a result, more taxis can be actuated for better sensing coverage quality with the same budget.

The ride requests in the city can be predicted based on the discovery that spatial and temporal ride request patterns tend to repeat on a weekly basis [177]. But even for the same city, ride request numbers vary across different days in a week, at different time periods in a day, and across different areas.

The ride request pattern in a city can be modeled as a time-evolving graph, called a ride request graph (RRG). For each time interval, a directed graph is constructed as follows. Each grid of the city containing the source or destination of a ride request is considered a node. Each source-destination pair is connected by a directed edge. The weight of the edge represents the ride request frequency between the same source and destination nodes. For each time interval t, the number of edges $e(t)$ and the number of nodes $n(t)$ follow the Densification Power Law (DPL):

$$e(t) = Kn(t)^{\gamma}, \tag{6.10}$$

where K and $\gamma \in [1, 2]$ are constant. The number of edges grows linearly according to the number of nodes if $\gamma = 1.0$, while the RRG becomes fully connected if $\gamma = 2.0$.

To predict the ride requests over time and location with the RRG, the system needs to learn two attributes: 1) the DPL factors (K and γ) which represent the temporal evolution property and 2) the spatial distribution of nodes in the RRG. The DPL factor can be calculated with the ride request history data used to construct the RRG. The spatial property can be obtained with the help of OSM Points of Interest (PoI) such as traffic signals, businesses, schools, hospitals etc [178], which are used to infer the ride request popularity in different areas. This model has been shown to be accurate in [177] by comparing it to real-world datasets. More details can also be found in [177].

6.3.4 Monetary Incentive

The key idea of my monetary incentive design is to include the probability of getting a second ride request at the actuated taxi's planned destination. This can decrease the monetary cost for actuating taxis by utilizing the underlying incentives of providing taxis a higher chance to get passengers at the destination of their assigned task. In this way, I could actuate more taxis and better utilize the budget to improve the sensing coverage quality.

The difference between taxis' distribution and the ride request distribution makes it possible to provide taxis with a higher chance of finding passengers in sparsely sensed areas. Therefore, if I could send the vehicles to the sparsely sensed areas with greater ride request probabilities, the cost for actuating taxis would be decreased and quality of sensing coverage would be improved. Meanwhile the utilities of the taxis are ensured, and overall transportation efficiency is improved. Therefore I design the monetary incentive $B(c)$ offered to taxi c as follows

$$B(c) = \max(r_{max} - r_u \cdot Request(x_c^T, y_c^T, T), r_{min}) \tag{6.11}$$

where r_{min} and r_{max} are the minimum and maximum monetary incentive to actuate one taxi respectively. This definition of the monetary incentive is based on the following reasons. First, the maximum incentive r_{max} should equal the maximum cost that the taxi incurs by following my route. Thus, I can find r_{max} from the gas, time cost and passenger count of driving during the actuation period. I can offer lower incentives, however, if taxis encounter ride requests while following my

trajectories: taxis could then earn additional money from serving these requests, which lowers their net cost from following my route. The $Request(i, j, t)$ represents the predicted ride request distribution in location of (i, j) at the t time interval, which is estimated using the model introduced in Section 6.3.3. r_u is the unit monetary incentive for one ride request. Moreover, even with a high possibility of getting a new ride request, each taxi still needs a minimum monetary incentive to motivate, which is r_{min}.

6.3.5 Multi-Incentive Taxis and Trajectory Selection Algorithm

To solve the NP-hard optimization problem in Eq.(2) (7), I propose a fast, near-optimal heuristic-based algorithm to find an approximate solution. The core idea of my algorithm is to 1) find times and locations with many taxis passing through, and 2) dispatch these taxis to different times and locations with very few taxis passing through. This is because actuating the taxis in a sparse area does not solve the problem of most taxis gathering in dense areas. The idea of the proposed algorithm is based on the Complementary Constructive Procedure (CCP). I first initialize a feasible solution S under the constraints, which is easy to implement by selecting taxis until the budget is full. Then I keep updating the solution to improve the objective function, which is the sensing quality. As Algorithm 5 shows, in the initialized feasible solution, I can find the corresponding time and location pair that contains maximum data points. I can also find the set S_{tmp} of taxis that pass through the location at the respective time. A key step is then that for taxis belonging to the set S_{tmp}, the expectation of the current trajectories is computed based on the current data point distribution. In this way, I can have an overall idea about which taxis passing through the maximum grid contribute the least to the overall data distribution balance level. Then I rank this list of taxis from least to most contribution in order to select taxis to dispatch to a new trajectory or not actuate. Similarly, I first compute the expectation value of each prospective trajectory, including random run without actuation, on the current data distribution. Then the algorithm traverses the prospective trajectories according to descending expectation value until the sensing coverage quality is improved. Finally, the solution is updated based on the selected trajectory and taxi. With multiple iterations, the solution keeps updating until the estimate of sensing coverage quality converges.

Algorithm 5: Multi-Incentive Algorithm for Taxis and Trajectory Selection

Input: Current location x_0, Budget R, Taxis availability, Ride request model *Request*, Mobility prediction model P
Output: Actuated taxis ID, planned trajectory and monetary incentive for actuated taxis
Initialize:
    Select taxis and trajectory randomly until the budget is full
    Output the initial feasible solution S based on actuated taxis and P for non-actuated taxis
while ϕ *converges* **do**
    Select the grid with maximum taxis passing through
    Take out the set of taxis S_{tmp} which pass through the maximum grid
    Compute and rank the contribution of trajectories of taxis belonging to S_{tmp}
    Select the taxi with minimum contribution and update its trajectory with monetary incentive defined by *Request*
    Keep updating the trajectory until the budget constraint R is satisfied
    Update S and calculate the updated sensing quality ϕ
end
Return $S^\star = S$

6.4 Evaluation

In this section, I evaluate my system's ability to achieve optimized sensing coverage quality with the same type platform in Chapter 4. In addition, I also verify its ability to match ride requests with taxis, which is an essential actuation motivation for taxis. I first introduce how I design a simulation based on real historical taxi trajectory data and real experiments on a taxi testbed for evaluation in Section 6.4.1. Then, I present and analyze the simulation and experiment results in Sections 6.4.2 and 6.4.3 respectively.

The evaluation focuses on the following aspects:

- Validating the effectiveness of integrating the mobility prediction model and ride request prediction model. This is done by comparing the performance of my system with several baselines, which will be introduced in Section 6.4.1 and analyzed in Section 6.4.2 and 6.4.3
- Evaluating the performance of the proposed algorithm on optimizing the sensing coverage quality and matching ride request. This is shown by the

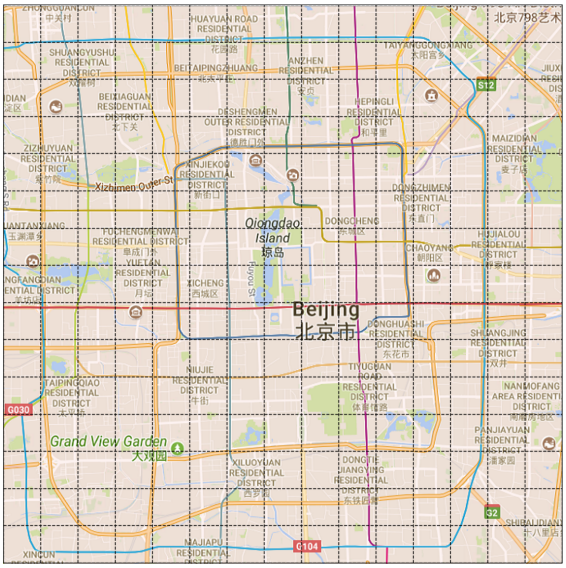


Figure 6.3: I evaluate my system in the center area of Beijing, which occupies a size of $15km$ by $15km$. The evaluation area is discretized into grids of $1km$ by $1km$.

metric performance defined in Section 6.4.1.

- Characterizing the system performance under different system set ups. Two key factors are discussed: budget amount (Figure 6.6 and 6.8) and total taxi number (Figure 6.7 and 6.9).

6.4.1 Evaluation Setup

I evaluate my system on a taxi testbed as well as on a simulation based on real historical taxi trajectory data in the center area of Beijing. The evaluation area occupies a size of $15km$ by $15km$, as shown in Figure 5.5. The major setup parameters of the evaluations are listed below.

Real Taxi Testbed Experiment Setup: To test my system in a realistic setting, I recruited taxis to run in the city of Beijing. I evaluate my system at four different times of day six hours apart: 0:00am, 6:00am, 12:00pm and 6:00pm. In addition, I also evaluate the system at 9:00am since it is a peak time in a day. I run the taxis on routes calculated by my system. For each route, a researcher hailed a taxi. The researcher suggested routes for the driver based on my system outputs. The drivers are free to modify routes. During the whole process, as shown in Figure 6.4, an Android App named GPS Logger was used to collect real-time GPS taxi data [1].

Figure 6.4: I evaluate my system with a real taxi testbed. An Android app, GPS Logger [1], is used to collect real-time GPS data.

In total, I collected traces from 230 actuated taxis over a period of 14 days. The experiment was approved under the university IRB *STUDY2017_00000342*.

Historical Trajectory Data Description: I use the Beijing taxi trajectory dataset for November 2015 [179] to conduct simulations based on real historical taxi trajectory data. The dataset is formatted as follows: taxi id, time stamp, longitude, latitude, occupancy flag. The occupancy flag represents whether the taxi is occupied by customers. The temporal and spatial resolutions are 60 seconds and 1 meter respectively. I extract the ride requests in the city according to the occupancy flag transformations. A ride request is obtained when a taxi's occupancy flag is turned to occupied.

General System Setup: Every actuation period, I randomly select 500 active taxis as the total vehicle fleet. I take temporal and spatial resolution as 2 minutes and 1 km since the average taxi speed in Beijing is 30km/h, and at this speed 2 minutes covers 1 km, which is one grid. The incentive in my system is given in units of US dollars (USD). I adopt $r_u = 2(USD)$, $r_{min} = 2(USD)$ and $r_{max} = 20(USD)$, since 2 USD is the flag-down fare of Beijing Taxi and 20 USD is enough to cover the cost for one trajectory ($\sim 10km$) in one incentive period under bad traffic conditions. I take the first 3 weeks' data to train mobility prediction and ride request prediction and the final days of the month to test the system. I evaluate my system every six hours, at 0:00am, 6:00am, 12:00pm and 6:00pm as well as at 9:00am. In the taxi testbed, the actuated taxis run on real roads as described earlier. In the simulated experimentation, I assume that the actuated taxis follow the planned trajectories at

an average velocity and finish the tasks before the end of one incentive period.

Performance Metric: Two metrics are introduced to show the performance of the algorithm in improving sensing coverage quality. The first one is the **value of sensing coverage quality (SCQ)** ϕ, which is the objective of my problem as shown in Eq 6.1.

Since it is difficult to understand whether the sensing coverage quality ϕ is good or not with a value, I adopt the **sensing coverage quality improvement percentage** ($SCQIP$) compared to the upper bound of sensing coverage quality as the performance metric. This metric evaluates how close the sensing coverage quality is to the ideal-maximum sensing coverage quality given the sensing coverage quality before actuation.

To be more specific, denoting the ideal-maximum sensing coverage quality as ϕ_{ideal}, the sensing coverage quality before actuation as ϕ_0, and improved sensing coverage quality obtained from current algorithm as $\phi^\star$, the improvement percentage of the current algorithm is

$$SCQIP = \frac{\phi^\star - \phi_0}{\phi_{ideal} - \phi_0} \tag{6.12}$$

$$\phi_{ideal} = \alpha \log(T \cdot N_{lon} \cdot N_{lat}) + (1 - \alpha) \log(T \cdot C) \tag{6.13}$$

It is noticed that $SCQIP$ cannot reach 100% since ϕ_{ideal} is the quality of sensing coverage in the ideal scenario. In the ideal case, the taxis' distribution is exactly uniform at each time point. However, because of limitations on initial locations of vehicles, this ideal case is impossible to achieve in practice.

Another important evaluation metric is the ride request matching rate. Matching ride requests for taxis is an essential factor in decide whether the drivers have enough motivation for actuation tasks. A matched request means that there is a ride request in the same grid square as the taxi destination once the taxi arrives. I define the **ride request matching rate (RRMR)** RR to evaluate the performance of the algorithm in improving ride request matching.

$$RR = \frac{\#\text{Actuated vehicle matched passengers at time } T}{\#\text{Actuated Vehicle}} \tag{6.14}$$

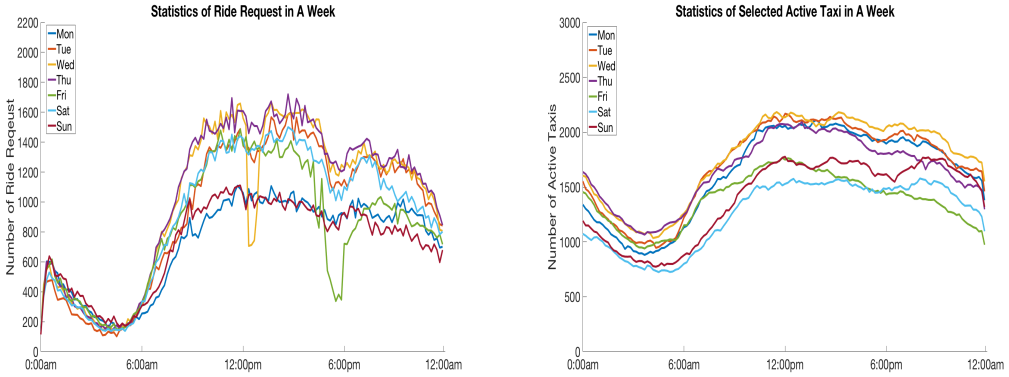
RR indicates how many actuated free-load vehicles can find a passenger at their

system-assigned destination. This performance metric evaluates the quality of the overall ride request matching.

Baselines: I adopt different baselines to validate different parts of my system in improving sensing coverage quality. These parts include mobility prediction model, ride request prediction model and the core algorithm.

- *No Actuation (NA)*: This method does nothing to actuate taxis or match ride requests. All the taxis just follow their original trajectories. By comparing this method with my ASC system, I can check the performance improvement of my entire system.
- *Random Actuation (RND)*: This method randomly selects taxis and routes to actuate taxis within the given budget. RND always offers the maximum monetary incentive. By comparing this method with my ASC system, I can check the performance improvement caused by the two prediction models.
- *Random Actuation with Ride Request Prediction (RND_RQ)*: This method also randomly selects taxis and actuation routes within the given budget. At the same time, *RND_RQ* tries to match ride requests with taxis. As a result, the cost to actuate one taxi will be lower than *Random Actuation (RND)*. By comparing this method with *RND*, I can check the improvement caused by the ride request model. By comparing this method with the ASC, I can check the improvement caused by the mobility prediction model.

Figure 6.5 shows the statistics of ride requests and active taxis in one week selected from my dataset. I calculate the number of ride requests and active taxis every 10 minutes. The active taxi number is calculated from a subset of around 3000 taxis. Ride request and active taxi counts show similar daily trends, corresponding to common supply and demand relations. Both show decreasing trends from 0:30am - 5:00am, when most people are asleep. After that, an increasing trend appears until 11:30am, as people go to work and school, do some shopping, etc. Both ride requests and active taxis maintain a high level from noon to midnight, which corresponds to the most busy time in Beijing.



(a) Ride requests within 10 minute time intervals on different days in a week.

(b) 10-minutes active taxi number on different days in a week.

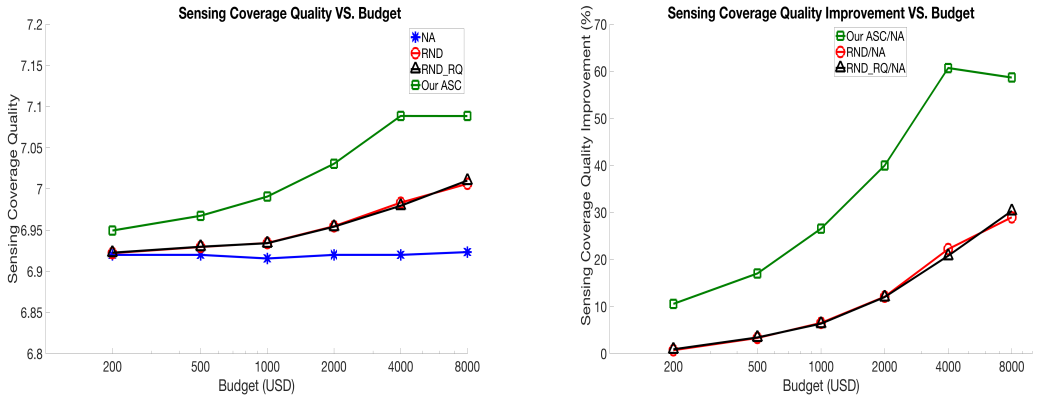
Figure 6.5: This figure shows the temporal distribution of ride requests and active taxis in a week. Active taxis have a similar trend as ride requests, which corresponds to human daily activity pattern in Beijing.

6.4.2 Physical Feature Based Simulation Performance

I conduct extensive simulations based on real historical taxi trajectory data to illustrate how my system optimizes sensing coverage quality and matches ride requests with taxis respectively. As discussed in Section 6.4.1, the results are obtained from average values of 5 time periods on 5 different days and compared with baselines.

I check the performance with two key factors: budget and total taxi number. Budget decides the amount of taxis that can be actuated. A larger budget usually means more actuated taxis for better sensing coverage quality. The total taxi number decides the amount of searching space the system can use for actuation. A larger total taxi number usually brings more candidate choices for the system to actuate, which leads to better sensing coverage quality.

Sensing Coverage Quality VS Budget: In order to evaluate how budget affects the sensing coverage quality with my ASC and baselines, I plot the sensing coverage quality and its improvement percentage IP in Figure 6.6. First, for ASC, RND and RND_RQ, sensing coverage quality improves with increasing budget. Higher budgets allow for more actuated taxis, leading to better sensing coverage quality. Second, my ASC always shows an advantage over the three baselines. Especially when the budget is 4000 USD, my ASC achieves 61% IP while RND and RND_RQ only give 22% and 20% IP over NA respectively. The 40% advantage of



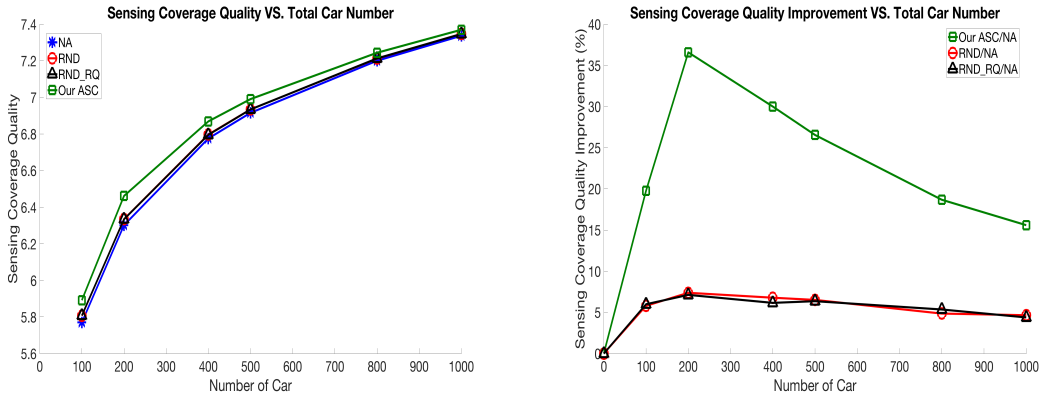
(a) Sensing Coverage Quality with Different Budgets

(b) Sensing Coverage Quality Improvement with Different Budgets

Figure 6.6: This figure shows the sensing coverage quality and its improvement with different budgets. My ASC always shows up to 40% more improvement than *RND* and *RND_RQ*. To achieve similar sensing coverage quality, My ASC needs 200 USD while *RND* and *RND_RQ* need 2000 USD.

my ASC has several causes. The ride request prediction model helps my ASC lower the incentive cost by matching ride requests with taxis. In addition, the mobility prediction model guides my ASC to select taxis which bring more sensing coverage quality improvement. Third, my ASC arrives at saturation point at 4000 USD while other baselines still keep increasing even at 8000 USD. This shows that with the help of two prediction models, my ASC is much more efficient at improving sensing quality coverage. To achieve similar sensing coverage quality, my ASC needs 200 USD while *RND* and *RND_RQ* need 2000, which is 10× my expense. Finally, although *RND_RQ* can lower incentive costs by matching more ride requests and thus actuating more taxis, it still does not exceed the sensing coverage quality of *RND*. This shows that even with more actuated taxis, randomly selecting taxis to actuate does not bring sensing coverage quality improvement. The similar trend of *RND* and *RND_RQ* validates the effect of my mobility prediction model.

Sensing Coverage Quality VS Number of Vehicles: The effect of vehicle number on sensing coverage quality is investigated with a fixed budget of 1000 units in simulation. Figure 6.7 shows the performance in improving sensing coverage quality compared to the baseline algorithm. In Figure 6.7(a), our ASC algorithm always performs better than baseline algorithms in all different numbers of vehicles.



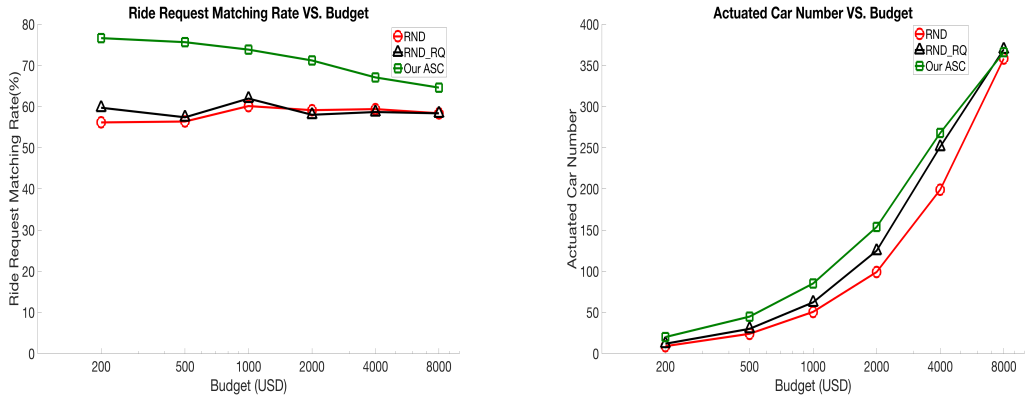
(a) Sensing Coverage Quality with Different Total Car Numbers

(b) Sensing Coverage Quality Improvement with Different Total Car Numbers

Figure 6.7: This figure shows the sensing coverage quality and its improvement with different total car numbers. Our ASC consistently show advantages on sensing coverage quality over baselines.

This proves that our ASC algorithm effectively selects "correct" vehicles with wise trajectories to get better sensing coverage quality. In Figure 6.7(a), the green line represents the increasing trend of sensing quality in the algorithm. Comparing the different algorithms, I see $ASC > RND \approx RND\_RQ > NA$. It is also shown that the sensing coverage quality increases with the number of vehicles under the same budget. Recall the definition of sensing coverage quality $\phi(A)$, which is the trade-off between the amount of collected data $Q(A)$ and data distribution balance level $E(A)$. For the amount of data $Q(A)$, more vehicles mean more collected data points. From the definition of data distribution balance level $E(A)$, intuitively, more vehicles would increase supply over demand and cause competition between vehicles. The competition forces some vehicles to drive to areas with fewer vehicles to increase their probability of getting passengers. Thus, the distribution of vehicles becomes more balanced and $E(A)$ increases with vehicle count. Therefore in Figure 6.7(a), the sensing coverage quality increases in both baseline algorithms and my algorithm. But my ASC algorithm still outperforms the baseline algorithms, showing the robustness of my algorithm under varying uncertainties in taxi availability.

Meanwhile, my ASC algorithm always achieves a higher improvement of sensing coverage quality compared to the other algorithms. It is already shown that non-actuation performs the worst in Figure 6.7(a). The non-actuation policy is taken as



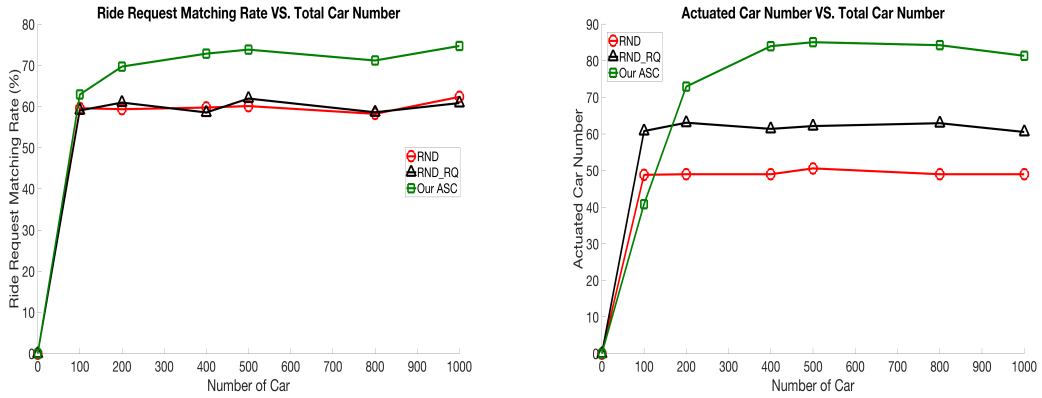
(a) Ride Request Matching Rate with Different Budgets

(b) Actuated Taxi Number with Different Budgets

Figure 6.8: This figure shows ride request matching rate and actuated taxi number with different budget. My ASC consistently shows up to 20% matching rate than *RND* and *RND_RQ*.

a reference to help evaluate how much better other algorithms perform in similar conditions. In Figure 6.7(b), the x-axis is the number of vehicles, and the y-axis is the improvement percentage (IP) of sensing quality using new algorithms compared to non-actuation. The green line represents my ASC algorithm. The red and black lines represent the random actuation and ride request-based random actuation algorithms. The ASC algorithm achieves up to 36.58% improvement with 200 vehicles, which is **5.12X** and **4.96X** higher than the performance of RND and RND_RQ algorithms respectively.

Ride Request Matching VS Budget: To evaluate how budget affects the ride request matching of different methods, I plot ride request matching rate and actuated taxi number in Figure 6.8. First, a large budget leads to more actuated taxis for all methods, but does not ensure large ride request matching rate, which is different from sensing coverage quality. This is because the first priority of my ASC is to improve sensing coverage quality. To ensure sensing coverage quality improvement, ASC will sacrifice ride request matching. Second, for different budgets, my ASC has up to a 20% larger ride request matching rate and than *RND* and *RND_RQ*. This shows that even though my ASC sacrifices ride request matching rate to guarantee sensing coverage quality improvement, it still keeps a higher matching rate than other methods.

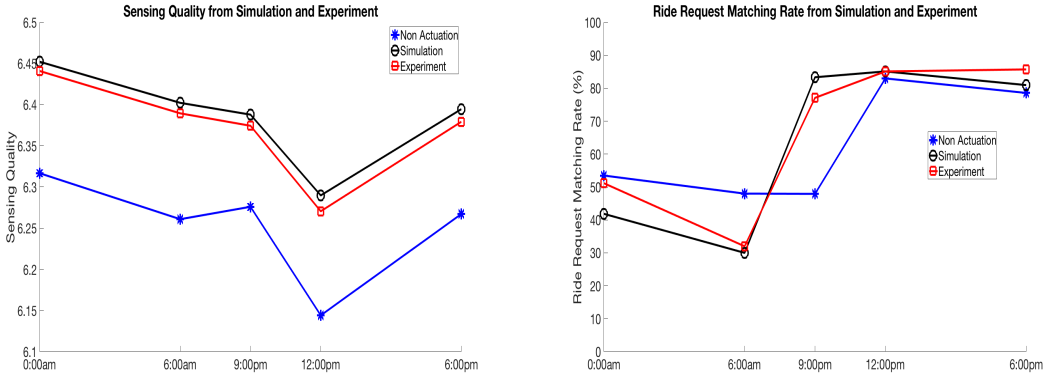


(a) Ride Request Matching Rate with Different Total Car Numbers

(b) Actuated Taxi Number with Different Total Car Numbers

Figure 6.9: This figure shows ride request matching rate and actuated taxi number with different total car numbers. My algorithm is robust to varying total vehicle numbers in matching ride requests.

Ride Request Matching VS Number of Vehicle: The effect of the number of vehicles on ride request matching rate and actuated taxi count is evaluated with a fixed budget of 1000. The results are shown in Figure 6.9. Figure 6.9(a) shows that my ASC algorithm achieves higher ride request matching rates with any number of vehicles. Figure 6.9(b) shows that my ASC algorithm actuated more vehicles within the same budget compared to other algorithms. Combining the results of these two figures, the ASC algorithm does help more vehicles improve the possibility of finding passengers. Figure 6.9(a) shows the variation of ride request matching rate under different vehicle counts. The green line represents the results obtained from my ASC algorithm. The blue, red, and black lines represent the results from non-actuation, random actuation and ride request-based random actuation respectively. It is shown that the ride request matching rate does not vary a lot with different vehicle counts for all algorithms. My ASC algorithm achieves up to 73.82% ride request matching rate with 500 vehicles, which is 11.59% and 13.71% higher than RND_RQ and RND. My algorithm is also robust to varying the total number of vehicles in matching ride requests. Figure 6.9(b) shows the actuated vehicle count under different total numbers of vehicles. With a total vehicle count of 200, the number of actuated vehicles is consistently higher using my ASC algorithm than with baseline algorithms. This proves that my algorithm could actuate more vehicles by using the budget



(a) Sensing Coverage Quality of Experiments at Different Times

(b) Ride Request Matching Rate of Experiments at Different Time

Figure 6.10: (a)The figure shows the sensing coverage quality from real experiment, physical feature based simulation and non-actuation. Experiment results show improvements similar to but slightly lower than simulation results. Both of them show advantages in sensing coverage quality at different times; (b)The figure shows the ride request matching rate from real experiments, physical feature based simulation, and non-actuation. Experiment results are similar to simulation results.

in an efficient way. Furthermore, the number of actuated vehicles converges to around 85 when the total number of vehicles equals 400 because of the limited budget. Nonetheless, increasing the number of vehicles still improves data coverage by providing more flexibility of vehicles and forcing the vehicles to run sparsely to avoid competition.

6.4.3 Experiment Results

To check the performance of my system in real operational conditions, I conducted experiments on a taxi based testbed at 5 representative time periods mentioned in Section 6.4.1. My experimental evaluation accounts for real-time traffic patterns, which the simulation does not. I compare the ASC experiment results with ASC simulation results to illustrate that my system is practical. In addition, I also utilize the taxi testbed to compare sensing coverage quality and ride request matching rates between ASC simulation and real experiment results. In addition, I include the non-actuation results as a baseline.

Figure 6.10(a) shows the sensing coverage quality from the real experiment,

physical feature based simulation and non-actuation. At all representative times, sensing coverage quality values from the experiment are similar to simulated values, which shows that physical feature based simulation can be used to analyze system operation in the real world. It is noticed that sensing coverage quality values from the experiment are a little bit lower than those from the simulation. This is because simulation results are theoretically near-optimal while real experiments involve practical factors that prevent it from achieving simulation results. These factors include traffic jams, temporary road closure, lack of direct routes to follow the designed trajectories, etc. In addition, both simulation and experiment sensing coverage quality show advantages over non actuation results. This proves that my system improves sensing coverage quality in both the simulation and experiment.

Figure 6.10(b) shows the ride request matching rate from the real experiment, physical feature based simulation and non-actuation. The similar ride request matching rate at all representative times in both the experiment and simulation proves that physical feature based simulation can be used to analyze system operations in the real world. Unlike sensing coverage quality results, simulation results for ride request matching rates are not always higher than experiment results. This is because optimizing sensing coverage quality is the first priority of my system. The optimal ride request matching rate cannot be achieved at the same time. In addition, at 0:00am and 6:00am, the non-actuation scheme has higher ride request matching rate than the ASC simulation and experiment results. When there are not many ride requests in the area (0:00am and 6:00am), the system has to choose the actuation routes that optimize sensing coverage quality but reduce ride request matching. When there are many ride requests in the area (9:00am, 12:00pm and 6:00pm), the system is able to choose the actuation routes that optimize sensing coverage quality as well as matching ride requests.

6.5 Related Work

In this section, I give a literature review of related work on two topics, *i.e.*, taxi behaviors & incentives and mobile crowdsensing.

6.5.1 Taxi Behaviors and Incentives

Taxis play an irreplaceable role in a city's transportation system by providing reliable and customized travel services for passengers. Compared with other transportation modes such as subway and bus services, taxis have no fixed routes, making them flexible and accessible from almost every corner of a city.

For the safety of passengers, taxis are required to be equipped with GPS trackers by law in many countries. Smart phones also make it easy to record the location of taxis. These GPS trajectories can be regarded as digital footprints of human mobility. Based on open taxi GPS datasets including Geolife [180], previous studies have addressed various research topics, including road map making [181, 182], urban mobility understanding [183, 184, 185, 186, 187], city region function identification [188, 189, 190], and location-based social networks [191].

Due to uncertain and time-variable traffic and ride request demand, mobility prediction and ride request prediction are still two challenging tasks for researchers. Human mobility is believed to have limits of predictability [192]. However, with more available data and the usage of state-of-art machine learning and deep learning models, the prediction accuracy of taxi mobility has been improved remarkably in the past few years [193]. Previous studies have also shown that ride requests follow the well-known densification power law, which may be used to predict or even synthesize ride requests [177].

Because drivers prioritize individual profit, taxis are unevenly distributed in different areas and overall efficiency is heavily harmed by competition in over-supplied areas and supply-demand imbalance in under-supplied areas. To help to improve the performance of taxi drivers and shorten the waiting time of passengers, online taxi-hailing service [194] and dynamic taxi dispatch system [195] are proposed to improve the scheduling efficiency of taxis. Ride-sharing services [196] are also proposed to increase the delivery capacity.

6.5.2 Mobile Crowdsensing

Crowdsensing, also called participatory sensing or community sensing, is collectively sharing data and extracting information to measure and map phenomena of common interest by individuals with sensing and computing devices [17]. As smart phones become more powerful and equipped with GPS trackers and accelerometers, crowd

sensing is becoming widely adopted as a flexible and low-cost method of collecting sensing data.

Usually the system's objective is to maximize sensing quality, which might have different metrics in different studies, (*e.g.*, k-depth coverage [197]), and the system may have different constraints, (*e.g.*, budget of rewards for participants). Sensing coverage is a major metric of evaluating sensing quality and has been used in many previous studies, including place-centric crowdsensing [198] and people-centric sensing [199]. While there are some previous studies that aim to maximize sensing quality under budget constraints [200], they design the scheme from a systematic view and do not consider the motivations of users, who have their own priorities and may try to game the system.

Auction-based mechanisms and game-theoretical models, *e.g.*, reverse auction [201] and Stackelberg game [170], are used to fix this problem. Furthermore, budget-feasible mechanisms [202] and proportional share rule-based compensation determination schemes [203] are proposed to guarantee strategy-proofness and budget feasibility. More discussion about auction-based mechanisms, as well as other incentive mechanisms, which may include lotteries, trust and reputation systems, can be found in previous surveys [165, 204, 205].

Auction-based incentive mechanisms can be well designed to possess desirable theoretical properties. In real implementations, the strong assumptions of participants being rational and strategic, the obscure theories, and the complex payment rules make them less attractive and practical. The time sensitivity of allocating sensing tasks also make it less likely that the participants will think about every possible situation and give a bid that accurately reflects their utility.

6.6 Conclusion

This chapter presented ASC, a system based on three types of spatiotemporal relationship that actuates vehicular taxi fleets for optimal sensing coverage quality by matching ride requests with taxis. I proposed a near-optimal algorithm that integrates 1) a mobility prediction model that guides the selection of which taxis to actuate and 2) a ride request prediction model to help match ride requests with taxis, lower incentive costs, and improve taxi drivers' motivation. Extensive simulation and

experiments on taxi testbeds show that ASC can achieve up to 40% improvement in sensing coverage quality and up to a 20% higher ride request matching rate than the baselines. Additionally, ASC can achieve similar sensing coverage quality as baseline algorithms with only 10% of the budget requirement. All these results validate the effectiveness of applying spatiotemporal relationships on the semi-controllable large scale mobile CPS for route planning, which involves system status and physical field estimation.

Chapter 7

Conclusions

This thesis presents a spatiotemporal relationship-aided framework for large scale mobile CPSs, as described in Cahpter 2, which includes an *spatiotemporal relationship analysis layer* to improve both field estimation (system status & physical field) and route planning on 3 types of systems. The improvement comes from three different types of spatiotemporal relationships. 1) The **spatiotemporal relationship of physical field (ST-PHY)**: the evolution of the physical field over space and time, determined by the laws of physics, indicates the relationship between values at adjacent areas and time periods, such as air pollution levels in a city across a day. 2) The **spatiotemporal relationship of system status (ST-SYS)**: system status changes continuously over time and space for the same device and the system status of the same type of devices may have similar values at the same areas and time periods. 3) The **spatiotemporal relationship connecting physical field and system status (ST-PHY-SYS)**: the physical field and system status are interrelated through time and space because they both react to the physical world. The research validates the proposed framework through prototypes of combinations of the system task and the system type. Two prototypes are about system status estimation (Chapter 3) and physical field estimation (Chapter 4) when the system does not control the routes of mobile devices. Another two prototypes are about route planning for controllable (Chapter 5) and semi-controllable (Chapter 6) systems to improve density and evenness of data distribution. Four prototypes prove that the *spatiotemporal relationship analysis layer* is able to address the challenges of sparse coverage and uneven distribution of data collection, as described in Chapter 1. The

thesis provides principles and guidelines in the design of uncontrollable, controllable and semi-controllable large scale mobile CPSs.

In particular, as shown in Chapter 3, the thesis contributes to system status estimation as follows.

- It is the first to investigate the context-aware App-usage prediction problem over a large user population. It considers context information (time & location), attribute information (mobile App type) and dynamic user preference.
- It finds that the relationships between App-location, App-time, and App-App type are essential to prediction and proposes a heterogeneous graph embedding algorithm to map them into one common comparable latent space. A user profile with personal App usage & trajectory history affected by a time decay factor is proposed to achieve a personalized prediction. Both the common attribution of all users and individual user dynamic preferences are extracted to ensure sufficient training data without losing personalization.
- It evaluates the system through a large-scale real-world dataset, which includes more than 6 million mobile App usage records from 1788 individual users. *CAP* demonstrates a significant improvement in the prediction accuracy compared to baselines.

The thesis contributes to physical field estimation, as shown in Chapter 4, as follows.

- It reformulates a physics guided model for air dispersion state evolution estimation and combines it with a data driven model.
- It proposes an adaptive scheme to correct estimates from a physics guided model with estimates generated from a data driven model.
- It deploys and evaluates the system with using a Particle Filter structure on a large-scale vehicular sensing platform for large-scale evaluation.

As shown in Chapter 5, the thesis contributes to the route planning of a controllable drone based CPS as follows.

- An adaptive planning algorithm for navigation enables the swarm to collaboratively achieve up to 6× reduction in location estimation errors, and as much as 3× improvement in navigation success rate under the given time and accuracy constraints.

- A planning algorithm determines the quality of location estimations and uses it to adaptively plan node motion.
- Real MAV testbed experiments and large scale physical feature based simulations are performed using radio signatures collected from the physical world and empirically determined sensor noise models validating our assumptions. Drone swarms with both homogeneous and heterogeneous structures are tested.

The thesis contributes to the route planning of a semi-controllable vehicle based CPS (Chapter 6) as follows.

- It presents a system to optimize sensing coverage quality through collaboration with matching ride requests with taxis, which solves the challenge of conflicting goals between the vehicle fleet and MCS platform.
- It formulates the collaboration task and proposes a near-optimal algorithmic solution, which integrates 1) a mobility prediction model to guide selecting taxis to improve effectiveness on actuating and 2) a ride request prediction model to help match ride requests with taxis, lower incentive costs and improved taxi driver motivation, which solves both challenges.
- The system is evaluated with real city-scale deployment and history trajectory data (46 actuated on average, 500 vehicles total) in the city of Beijing, China.

Both real deployed testbed experiments and extensive simulations with real world collected data are adopted to validate the system design. As a part of the evaluation for uncontrolled to controlled motion aspects of our system, air pollution sensors are deployed on the taxi-based testbed to collect data in the city of Shenzhen for 2 years in collaboration with Tsinghua University. In addition, a swarm of 8 micro aerial vehicles are deployed in an indoor environment for autonomous navigation. The results show that incorporating the *spatiotemporal relationship analysis layer* can achieve 2.1× and 6× error reduction on physical field and system status estimation and 3× improvement on route planning. This shows that the proposed framework and algorithms have the potential to make large scale mobile CPSs more intelligent in sensing estimation and route planning. The work should provide a foundation for further research in the emerging area of large scale mobile CPSs.

Bibliography

[1] "Gps logger for android. 2017. google play shop," https://play.google.com/store/apps/details?id=com.mendhak.gpslogger&hl=en.

[2] Edward A Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2008, pp. 363–369.

[3] Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic, "Cyber-physical systems: the next computing revolution," in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 731–736.

[4] Beecham Research, "M2m/iot sector map," http://www.beechamresearch.com/article.aspx?id=4.

[5] National Institute of Standards and Technology, "Advanced cyber-physical systems for national priorities (+$7.5 million)," https://www.nist.gov/property-fieldsection/advanced-cyber-physical-systems-national-priorities-75-million.

[6] Marco Annunziata and Peter C Evans, "Industrial internet: Pushing the boundaries of minds and machines," *General Electric*, 2012.

[7] Behrad Bagheri, Shanhu Yang, Hung-An Kao, and Jay Lee, "Cyber-physical systems architecture for self-aware machines in industry 4.0 environment," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 1622–1627, 2015.

[8] Stefan Heng, "Industry 4.0: Huge potential for value creation waiting to be tapped," *Deutsche Bank Research*, 2014.

[9] David Sousa Nunes, Pei Zhang, and Jorge Sá Silva, "A survey on human-in-the-loop applications towards an internet of all," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 944–965, 2015.

[10] Hackernoon, "How much time do people spend on their mobile phones in 2017," https://hackernoon.com/how-much-time-do-people-spend-on-their-mobile-phones-in-2017-e5f90a0b10a6.

[11] Cisco Visual networking Index, "Forecast and methodology, 2016-2021, white paper," *San Jose, CA, USA*, 2016.

[12] BBC World News, "Air pollution causes 467,000 premature deaths a year in europe," http://www.bbc.com/news/world-europe-38078488.

[13] The New York Times Energy and Environment, "Study links 6.5 million deaths each year to air pollution," https://www.nytimes.com/2016/06/27/business/energy-environment/study-links-6-5-million-deaths-each-year-to-air-pollution.html.

[14] WHO Media centre, "Ambient (outdoor) air quality and health," http://www.who.int/mediacentre/factsheets/fs313/en/, 2014, Accessed: 2015-09-16.

[15] Z. Qian, R.S. Chapman, W. Hu, F. Wei, L.R. Korn, J. Zhang, et al., "Using air pollution based community clusters to explore air pollution health effects in children," *Environment international*, vol. 30, no. 5, pp. 611–620, 2004.

[16] A. Millman, D. Tang, and F.P. Perera, "Air pollution threatens the health of children in china," *Pediatrics*, vol. 122, no. 3, pp. 620–628, 2008.

[17] Raghu K Ganti, Fan Ye, and Hui Lei, "Mobile crowdsensing: current state and future challenges.," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[18] Bin Guo, Zhiwen Yu, Xingshe Zhou, and Daqing Zhang, "From participatory sensing to mobile crowd sensing," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*. IEEE, 2014, pp. 593–598.

[19] David Mage, Guntis Ozolins, Peter Peterson, Anthony Webster, Rudi Orthofer, Veerle Vandeweerd, and Michael Gwynne, "Urban air pollution in megacities of the world," *Atmospheric Environment*, vol. 30, no. 5, pp. 681 – 686, 1996, Supercities: Environment Quality and Sustainable Development.

[20] Aveek Purohit, Zheng Sun, Frank Mokaya, and Pei Zhang, "Sensorfly: Controlled-mobile sensing platform for indoor emergency response applications,"

in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE, 2011, pp. 223–234.

[21] Shenggong Ji, Yu Zheng, and Tianrui Li, "Urban sensing based on human mobility," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 1040–1051.

[22] "Smartphones industry: Statistics & Facts," `https://www.statista.com/topics/840/smartphones/`, 2017.

[23] "Mobile App Usage - Statistics & Facts," `https://www.statista.com/topics/1002/mobile-app-usage/`, 2017.

[24] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin, "Diversity in smartphone usage," in *Proc. ACM MobiSys*, 2010, pp. 179–194.

[25] Denzil Ferreira, Jorge Goncalves, Vassilis Kostakos, Louise Barkhuus, and Anind K Dey, "Contextual experience sampling of mobile application micro-usage," in *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*. ACM, 2014, pp. 91–100.

[26] Simon L Jones, Denzil Ferreira, Simo Hosio, Jorge Goncalves, and Vassilis Kostakos, "Revisitation analysis of smartphone app use," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015, pp. 1197–1208.

[27] Karen Church, Denzil Ferreira, Nikola Banovic, and Kent Lyons, "Understanding the challenges of mobile phone usage data," in *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 2015, pp. 504–514.

[28] Chunhui Zhang, Xiang Ding, Guanling Chen, Ke Huang, Xiaoxiao Ma, and Bo Yan, "Nihao: A predictive smartphone application launcher," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2012, pp. 294–313.

[29] Choonsung Shin, Jin-Hyuk Hong, and Anind K Dey, "Understanding and prediction of mobile application usage for smart phones," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 173–182.

[30] Denzil Ferreira, Eija Ferreira, Jorge Goncalves, Vassilis Kostakos, and Anind K Dey, "Revisiting human-battery interaction with an interactive battery interface," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 563–572.

[31] Vassilis Kostakos, Denzil Ferreira, Jorge Goncalves, and Simo Hosio, "Modelling smartphone usage: a markov state transition model," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 486–497.

[32] Ye Xu, Mu Lin, Hong Lu, Giuseppe Cardone, Nicholas Lane, Zhenyu Chen, Andrew Campbell, and Tanzeem Choudhury, "Preference, context and communities:a multi-faceted approach to predicting smartphone app usage patterns," in *Proc. ACM ISWC*, 2013, pp. 69–76.

[33] Vincent Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang, "Collaborative filtering meets mobile recommendation: A user-centered approach.," in *AAAI*, 2010, vol. 10, pp. 236–241.

[34] Hengshu Zhu, Enhong Chen, Kuifei Yu, Huanhuan Cao, Hui Xiong, and Jilei Tian, "Mining personal context-aware preferences for mobile users," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 1212–1217.

[35] Alexandros Karatzoglou, Linas Baltrunas, Karen Church, and Matthias Böhmer, "Climbing the app wall: enabling mobile app discovery through context-aware recommendations," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 2527–2530.

[36] Qi Liu, Haiping Ma, Enhong Chen, and Hui Xiong, "A survey of context-aware mobile recommendations," *International Journal of Information Technology & Decision Making*, vol. 12, no. 01, pp. 139–172, 2013.

[37] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts.," in *AAAI*, 2016, pp. 194–200.

[38] Niels van Berkel, Chu Luo, Theodoros Anagnostopoulos, Denzil Ferreira, Jorge Goncalves, Simo Hosio, and Vassilis Kostakos, "A systematic assessment of smartphone usage gaps," in *Proceedings of the 2016 CHI Conference on Human*

Factors in Computing Systems. ACM, 2016, pp. 4711–4721.

[39] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *International Conference on World Wide Web*, 2010, pp. 811–820.

[40] Duen-Ren Liu, Pei-Yun Tsai, and Po-Huan Chiu, "Personalized recommendation of popular blog articles for mobile applications," *Information Sciences*, vol. 181, no. 9, pp. 1552–1572, 2011.

[41] Enrique Costa-Montenegro, Ana Belén Barragáns-Martínez, and Marta Rey-López, "Which app? a recommender system of applications in markets: Implementation of the service for monitoring users' interaction," *Expert systems with applications*, vol. 39, no. 10, pp. 9367–9375, 2012.

[42] Matthias Böhmer, Lyubomir Ganev, and Antonio Krüger, "Appfunnel: A framework for usage-centric evaluation of recommender systems that suggest mobile applications," in *Proceedings of the 2013 international conference on Intelligent user interfaces*. ACM, 2013, pp. 267–276.

[43] Chen Lin, Runquan Xie, Xinjun Guan, Lei Li, and Tao Li, "Personalized news recommendation via implicit social experts," *Information Sciences*, vol. 254, pp. 1–18, 2014.

[44] Hal Abelson, Ken Ledeen, and Chris Lewis, "Just deliver the packets, in," *Essays on Deep Packet Inspection", Ottawa". Office of the Privacy Commissioner of Canada. Retrieved*, pp. 01–08, 2010.

[45] "China TeleCom Webpage," `http://www.chinatelecom-h.com/en/global/home.php/`, 2018.

[46] Hongyi Yao, Gyan Ranjan, Alok Tongaonkar, Yong Liao, and Zhuoqing Morley Mao, "Samples: Self adaptive mining of persistent lexical snippets for classifying mobile application traffic," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 2015, pp. 439–451.

[47] Derya Birant and Alp Kut, "St-dbscan: An algorithm for clustering spatial–temporal data," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 208–221, 2007.

[48] Joseph Turian, Lev Ratinov, and Yoshua Bengio, "Word representations: a

simple and general method for semi-supervised learning," in *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, 2010, pp. 384–394.

[49] David M Blei, Andrew Y Ng, and Michael I Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[50] Solomon Kullback and Richard A Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[51] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[52] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in neural information processing systems*, 2011, pp. 693–701.

[53] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang, "Learning graph-based poi embedding for location-based recommendation," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 15–24.

[54] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan, "Personalized ranking metric embedding for next new poi recommendation.," in *IJCAI*, 2015, pp. 2069–2075.

[55] Konrad Blaszkiewicz, Konrad Blaszkiewicz, Konrad Blaszkiewicz, and Alexander Markowetz, "Differentiating smartphone users by app usage," in *Proc. ACM Ubicomp*, 2016, pp. 519–523.

[56] Sha Zhao, Julian Ramos, Jianrong Tao, Ziwen Jiang, Shijian Li, Zhaohui Wu, Gang Pan, and Anind K. Dey, "Discovering different kinds of smartphone users through their application usage behaviors," in *Proc. ACM UbiComp*, 2016, pp. 498–509.

[57] Xiaoxing Zhao, Yuanyuan Qiao, Zhongwei Si, Jie Yang, and Anders Lindgren, "Prediction of user app usage behavior from geo-spatial data," in *Proc. ACM GeoRich*, 2016, pp. 1–6.

[58] Ke Huang, Chunhui Zhang, Xiaoxiao Ma, and Guanling Chen, "Predicting mo-

bile application usage using contextual information," in *Proc. ACM UbiComp*, 2012, pp. 1059–1065.

[59] Shenglin Zhao, Tong Zhao, Haiqin Yang, Michael R Lyu, and Irwin King, "Stellar: Spatial-temporal latent ranking for successive point-of-interest recommendation.," in *AAAI*, 2016, pp. 315–322.

[60] Upasna Bhandari, Kazunari Sugiyama, Anindya Datta, and Rajni Jindal, "Serendipitous recommendation for mobile apps using item-item similarity graph," in *Asia Information Retrieval Symposium*. Springer, 2013, pp. 440–451.

[61] Ning Chen, Steven CH Hoi, Shaohua Li, and Xiaokui Xiao, "Simapp: A framework for detecting similar mobile applications by online kernel learning," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 2015, pp. 305–314.

[62] Ning Chen, Steven CH Hoi, Shaohua Li, and Xiaokui Xiao, "Mobile app tagging," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 2016, pp. 63–72.

[63] Dragomir Yankov, Pavel Berkhin, and Rajen Subba, "Interoperability ranking for mobile applications," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 857–860.

[64] Hengshu Zhu, Chuanren Liu, Yong Ge, Hui Xiong, and Enhong Chen, "Popularity modeling for mobile apps: A sequential approach," *IEEE transactions on cybernetics*, vol. 45, no. 7, pp. 1303–1314, 2015.

[65] Kent Shi and Kamal Ali, "Getjar mobile application recommendations with very sparse datasets," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 204–212.

[66] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua, "Addressing cold-start in app recommendation: latent user models constructed from twitter followers," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 283–292.

[67] Bin Liu, Deguang Kong, Lei Cen, Neil Zhenqiang Gong, Hongxia Jin, and Hui Xiong, "Personalized mobile app recommendation: Reconciling app functionality and user privacy preference," in *Proceedings of the Eighth ACM*

International Conference on Web Search and Data Mining. ACM, 2015, pp. 315–324.

[68] Hengshu Zhu, Hui Xiong, Yong Ge, and Enhong Chen, "Mobile app recommendations with security and privacy awareness," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 951–960.

[69] Peifeng Yin, Ping Luo, Wang-Chien Lee, and Min Wang, "App recommendation: a contest between satisfaction and temptation," in *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 2013, pp. 395–404.

[70] Xinjiang Lu, Zhiwen Yu, Bin Guo, and Xingshe Zhou, "Predicting the content dissemination trends by repost behavior modeling in mobile social networks," *Journal of Network and Computer Applications*, vol. 42, pp. 197–207, 2014.

[71] Zhiyuan Cheng, James Caverlee, Kyumin Lee, and Daniel Z Sui, "Exploring millions of footprints in location sharing services.," *ICWSM*, vol. 2011, pp. 81–88, 2011.

[72] Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil, "An empirical study of geographic user activity patterns in foursquare.," *ICwSM*, vol. 11, pp. 70–573, 2011.

[73] Justin Cranshaw, Raz Schwartz, Jason I Hong, and Norman Sadeh, "The livehoods project: Utilizing social media to understand the dynamics of a city," 2012.

[74] Zhu Wang, Daqing Zhang, Xingshe Zhou, Dingqi Yang, Zhiyong Yu, and Zhiwen Yu, "Discovering and profiling overlapping communities in location-based social networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 4, pp. 499–509, 2014.

[75] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhu Wang, "A sentiment-enhanced personalized location recommendation system," in *Proceedings of the 24th ACM Conference on Hypertext and Social Media*. ACM, 2013, pp. 119–128.

[76] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhiwen Yu, "Fine-grained preference-aware location search leveraging crowdsourced digital footprints

from lbsns," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 479–488.

[77] Mohamed Sarwat, Ahmed Eldawy, Mohamed F Mokbel, and John Riedl, "Plutus: leveraging location-based social networks to recommend potential customers to venues," in *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*. IEEE, 2013, vol. 1, pp. 26–35.

[78] Dingqi Yang, Daqing Zhang, Longbiao Chen, and Bingqing Qu, "Nationtelescope: Monitoring and visualizing large-scale collective behavior in lbsns," *Journal of Network and Computer Applications*, vol. 55, pp. 170–180, 2015.

[79] Marc Olivier Killijian, "Next place prediction using mobility markov chains," in *The Workshop on Measurement, Privacy, and Mobility*, 2012, p. 3.

[80] Meng Chen, Yang Liu, and Xiaohui Yu, "Nlpmm: A next location predictor with markov modeling," vol. 8444, pp. 186–197, 2014.

[81] Wesley Mathew, Ruben Raposo, and Bruno Martins, "Predicting future locations with hidden markov models," in *ACM Conference on Ubiquitous Computing*, 2012, pp. 911–918.

[82] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T Hanratty, and J. Han, "Gmove: Group-level mobility modeling using geo-tagged social media," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1305–1314.

[83] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti, "Wherenext: A location predictor on trajectory pattern mining," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2009, KDD '09, pp. 637–646, ACM.

[84] Josh Jia-Ching Ying, Wang-Chien Lee, Tz-Chiao Weng, and Vincent S. Tseng, "Semantic trajectory mining for location prediction," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, New York, NY, USA, 2011, GIS '11, pp. 34–43, ACM.

[85] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi, "Trajectory pattern mining," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA,

2007, KDD '07, pp. 330–339, ACM.

[86] Noel De Nevers, *Air pollution control engineering*, Waveland Press, 2010.

[87] Yun Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, Yilong Li, Ji Jia, and Xiaofan Jiang, "Aircloud: a cloud-based air-quality monitoring system for everyone," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 2014, pp. 251–265.

[88] David Hasenfratz, Olga Saukh, Christoph Walser, Christoph Hueglin, Martin Fierz, Tabita Arn, Jan Beutel, and Lothar Thiele, "Deriving high-resolution urban air pollution maps using mobile sensor nodes," *Pervasive and Mobile Computing*, vol. 16, pp. 268–285, 2015.

[89] JC Weil and RP Brower, "An updated gaussian plume model for tall stacks," *Journal of the Air Pollution Control Association*, vol. 34, no. 8, pp. 818–827, 1984.

[90] Zahari Zlatev and Ivan Dimov, *Computational and numerical challenges in environmental modelling*, vol. 13, Elsevier, 2006.

[91] Paolo Zannetti, *Air Pollution Modeling: Theories, computational methods and available software*, Springer Science & Business Media, 2013.

[92] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh, "U-air: When urban air quality inference meets big data," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1436–1444.

[93] Yu Zheng, Xuxu Chen, Qiwei Jin, Yubiao Chen, Xiangyun Qu, Xin Liu, Eric Chang, Wei-Ying Ma, Yong Rui, and Weiwei Sun, "A cloud-based knowledge discovery system for monitoring fine-grained air quality," *MSR-TR-2014–40, Tech. Rep.*, 2014.

[94] Yidan Hu, Guojun Dai, Jin Fan, Yifan Wu, and Hua Zhang, "Blueaer: A fine-grained urban pm2. 5 3d monitoring system using mobile sensing," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[95] George Keith Batchelor, *An introduction to fluid dynamics*, Cambridge university press, 2000.

[96] Kane Yee, "Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media," *IEEE Transactions on antennas and propagation*, vol. 14, no. 3, pp. 302–307, 1966.

[97] S Pal Arya, *Air pollution meteorology and dispersion*, Oxford University Press New York, 1999.

[98] John M Stockie, "The mathematics of atmospheric dispersion modeling," *Siam Review*, vol. 53, no. 2, pp. 349–372, 2011.

[99] A. Doucet, J.F.G. de Freitas, and N.J. Gordon, Eds., *Sequential Monte Carlo methods in practice*, Springer-Verlag, 2001.

[100] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2006.

[101] Xinlei Chen, Aveek Purohit, Carlos Ruiz Dominguez, Stefano Carpin, and Pei Zhang, "Drunkwalk: Collaborative and adaptive planning for navigation of micro-aerial sensor swarms," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 295–308.

[102] Pei Zhang and Margaret Martonosi, "Locale: Collaborative localization estimation for sparse mobile sensor networks," in *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*. IEEE, 2008, pp. 195–206.

[103] Chao Zhang, Keyang Zhang, Quan Yuan, Haoruo Peng, Yu Zheng, Tim Hanratty, Shaowen Wang, and Jiawei Han, "Regions, periods, activities: Uncovering urban dynamics via cross-modal representation learning," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 361–370.

[104] Xianda Zhang, "Matrix analysis and applications," *Tsinghua and Springer Publishing house, Beijing*, pp. 71–100, 2004.

[105] Michael Grant, Stephen Boyd, and Yinyu Ye, "Cvx: Matlab software for disciplined convex programming," 2008.

[106] Michael C Grant and Stephen P Boyd, "Graph implementations for nonsmooth convex programs," in *Recent advances in learning and control*, pp. 95–110. Springer, 2008.

[107] Sun-Chong Wang, "Artificial neural network," in *Interdisciplinary computing*

in java programming, pp. 81–100. Springer, 2003.

[108] Claude Elwood Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[109] Xiangxiang Xu, Xinlei Chen, Xinyu Liu, Hae Young Noh, Pei Zhang, and Lin Zhang, "Gotcha ii–deployment of a vehicle-based environmental sensing system," in *Proceedings of the 14th ACM Conference on Embedded Networked Sensor Systems. ACM, 2016.* ACM, 2016.

[110] "Criteria air pollutants," `https://www.epa.gov/criteria-air-pollutants`.

[111] "Carbon monoxide levels & risks," `https://www.myhomecomfort.org/carbon-monoxide-levels-risks/`.

[112] "Who guidelines for indoor air quality: Selected pollutants.," `https://www.ncbi.nlm.nih.gov/books/NBK138710/`.

[113] "China national air pollution monitoring station history data publishing," `http://beijingair.sinaapp.com`.

[114] Aaron Van Donkelaar, Randall V Martin, and Rokjin J Park, "Estimating ground-level pm2.5 using aerosol optical depth determined from satellite remote sensing," *Journal of Geophysical Research: Atmospheres*, vol. 111, no. D21, 2006.

[115] LN Lamsal, RV Martin, A Van Donkelaar, M Steinbacher, EA Celarier, E Bucsela, EJ Dunlea, and JP Pinto, "Ground-level nitrogen dioxide concentrations inferred from the satellite-borne ozone monitoring instrument," *Journal of Geophysical Research: Atmospheres*, vol. 113, no. D16, 2008.

[116] Randall V Martin, "Satellite remote sensing of surface air quality," *Atmospheric Environment*, vol. 42, no. 34, pp. 7823–7843, 2008.

[117] N.S. Holmes and L. Morawska, "A review of dispersion modelling and its application to the dispersion of particles: An overview of different dispersion models available," *Atmospheric Environment*, vol. 40, no. 30, pp. 5902 – 5928, 2006.

[118] Enkeleida Lushi and John M. Stockie, "An inverse gaussian plume approach for estimating atmospheric pollutant emissions from multiple point sources,"

Atmospheric Environment, vol. 44, no. 8, pp. 1097 – 1107, 2010.

[119] Tsang-Chu Yu, Chung-Chih Lin, Ren-Guey Lee, Chao-Heng Tseng, and Shi-Ping Liu, "Wireless sensing system for prediction indoor air quality," in *High Speed Intelligent Communication Forum (HSIC), 2012 4th International*, May 2012, pp. 1–3.

[120] O.A. Postolache, J.M.D. Pereira, and P.M.B.S. Girao, "Smart sensors network for air quality monitoring applications," *Instrumentation and Measurement, IEEE Transactions on*, vol. 58, no. 9, pp. 3253–3262, Sept 2009.

[121] Mikhail Kanevski, *Advanced Mapping of Environmental Data/Geostatistics, Machine Learning and Bayesian Maximum Entropy*, Wiley-IEEE Press, 2008.

[122] Michael Sherman, *Spatial statistics and spatio-temporal data: covariance functions and directional properties*, John Wiley & Sons, 2011.

[123] A Purohit, Zheng Sun, F Mokaya, and Pei Zhang, "SensorFly: Controlled-mobile sensing platform for indoor emergency response applications," in *Proceeding of the 10th International Conference on Information Processing in Sensor Networks (IPSN)*, 2011, pp. 223–234.

[124] Robert J. Wood, "The First Takeoff of a Biologically Inspired At-Scale Robotic Insect," *IEEE transactions on robotics*, vol. 24, no. 2, pp. 341–347, 2008.

[125] Karthik Dantu, Bryan Kate, Jason Waterman, Peter Bailis, and Matt Welsh, "Programming micro-aerial vehicle swarms with karma," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2011, SenSys '11, pp. 121–134, ACM.

[126] Matthew Turpin, Nathan Michael, and Vijay Kumar, "Trajectory design and control for aggressive formation flight with quadrotors," *Autonomous Robots*, vol. 33, no. 1-2, pp. 143–156, Feb. 2012.

[127] S. Shen, N. Michael, and V. Kumar, "Vision-based autonomous navigation in complex environments with a quadrotor," in *iros*, Tokyo, Japan, nov 2013.

[128] Ryan Morlok and Maria Gini, "Dispersing robots in an unknown environment," in *7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2004.

[129] Aveek Purohit, Zheng Sun, and Pei Zhang, "Sugarmap: Location-less coverage

for micro-aerial sensing swarms," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, New York, NY, USA, 2013, IPSN '13, pp. 253–264, ACM.

[130] I. Constandache, R.R. Choudhury, and I. Rhee, "Towards mobile phone localization without war-driving," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1–9.

[131] Zheng Sun, Aveek Purohit, Shijia Pan, Frank Mokaya, Raja Bose, and Pei Zhang, "Polaris: Getting accurate indoor orientations for mobile devices using ubiquitous visual patterns on ceilings," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, New York, NY, USA, 2012, HotMobile '12, pp. 14:1–14:6, ACM.

[132] Johann Borenstein, Liqiang Feng, and H. R. Everett, *Navigating Mobile Robots: Systems and Techniques*, A. K. Peters, Ltd., Natick, MA, USA, 1996.

[133] Aveek Purohit, Pei Zhang, Brian M Sadler, and Stefano Carpin, "Deployment of swarms of micro-aerial vehicles: from theory to practice," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5408–5413.

[134] Digi International, "XBee Multipoint RF Modules Datasheet," 2009.

[135] A Purohit and Pei Zhang, "Controlled-mobile sensing simulator for indoor fire monitoring," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, 2011, pp. 1124–1129.

[136] Maxim A Batalin and Gaurav S Sukhatme, "Coverage, exploration, and deployment by a mobile robot and communication network," in *Information Processing in Sensor Networks*. Springer, 2003, pp. 376–391.

[137] Maxim A Batalin and Gaurav S Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems*, vol. 26, no. 2-4, pp. 181–196, 2004.

[138] Ting Liu, Christopher M Sadler, Pei Zhang, and Margaret Martonosi, "Implementing software on resource-constrained mobile sensors: Experiences with impala and zebranet," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services*. ACM, 2004, pp. 256–269.

[139] Yong Wang, Pei Zhang, Ting Liu, Chris Sadler, and Margaret Martonosi, "Movement data traces from princeton zebranet deployments," *CRAWDAD Database*, vol. 129, 2007.

[140] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme, "An Incremental Self-Deployment Algorithm for Mobile Sensor Networks," *Autonomous Robots*, vol. 13, no. 2, pp. 113–126, Sept. 2002.

[141] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*, pp. 299–308. Springer, 2002.

[142] Gaurav S. Sukhatme Maxim A. Batalin, "Spreading Out: A Local Approach to Multi-robot Coverage," *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics System*, 2002.

[143] Wolfram Burgard, Cyrill Stachniss, Giorgio Grisetti, Bastian Steder, Rainer Kummerle, Christian Dornhege, Michael Ruhnke, Alexander Kleiner, and Juan D Tardós, "A comparison of slam algorithms based on a graph of relations," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 2089–2095.

[144] Denis Chekhlov, Mark Pupilli, Walterio Mayol-Cuevas, and Andrew Calway, "Real-time and robust monocular slam using predictive multi-resolution descriptors," in *Advances in Visual Computing*, pp. 276–285. Springer, 2006.

[145] Austin I Eliazar and Ronald Parr, "Dp-slam 2.0," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. IEEE, 2004, vol. 2, pp. 1314–1320.

[146] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot, "Consistency of the ekf-slam algorithm," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 3562–3568.

[147] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba., "A solution to the simultaneous localisation and map building (SLAM) problem," *IEEE Transactions of Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[148] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis, "Analysis

and improvement of the consistency of extended kalman filter based slam," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 473–479.

[149] Shoudong Huang and Gamini Dissanayake, "Convergence and consistency analysis for extended kalman filter based slam," *Robotics, IEEE Transactions on*, vol. 23, no. 5, pp. 1036–1049, 2007.

[150] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 2432–2437.

[151] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 36–46, 2007.

[152] Giorgio Grisetti, Gian Diego Tipaldi, Cyrill Stachniss, Wolfram Burgard, and Daniele Nardi, "Fast and accurate slam with rao–blackwellized particle filters," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 30–38, 2007.

[153] Andrew Howard, "Multi-robot simultaneous localization and mapping using particle filters," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[154] P. Bahl and V. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *IEEE Int. Conference on Computer Communications*, 2000, pp. 775–784.

[155] A. Ladd, K. Bekris, A. Rudys, D. Wallach, and L. Kavraki, "On the feasibility of using wireless ethernet for indoor localization," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 555–559, 2004.

[156] Moustafa Youssef and Ashok Agrawala, "The horus wlan location determination system," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 205–218.

[157] S. Zickler and M. Veloso, "RSS-based relative localization and tethering for moving robots in unknown environments," in *IEEE Int. Conference on Robotics and Automation*, 2010, pp. 5466–5471.

[158] J. Twigg, J. Fink, P.L. Yu, and B.M. Sadler, "Efficient base station connectivity area discovery," *International Journal of Robotics Research*, 2013.

[159] Brian Ferris, Dieter Fox, and Neil Lawrence, "Wifi-slam using gaussian process latent variable models," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, San Francisco, CA, USA, 2007, IJCAI'07, pp. 2480–2485, Morgan Kaufmann Publishers Inc.

[160] Mikkel Baun Kjærgaard, "A taxonomy for radio location fingerprinting," in *Location-and Context-Awareness*, pp. 139–156. Springer, 2007.

[161] Guobin Shen, Zhuo Chen, Peichao Zhang, Thomas Moscibroda, and Yongguang Zhang, "Walkie-markie: Indoor pathway mapping made easy," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, Berkeley, CA, USA, 2013, nsdi'13, pp. 85–98, USENIX Association.

[162] Jie Yin, Qiang Yang, and Lionel M Ni, "Learning adaptive temporal radio maps for signal-strength-based location estimation," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 7, pp. 869–883, 2008.

[163] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, New York, NY, USA, 2012, MobiSys '12, pp. 197–210, ACM.

[164] Aveek Purohit, Zheng Sun, Shijia Pan, and Pei Zhang, "Sugartrail: Indoor navigation in retail environments without surveys and maps," in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on*. IEEE, 2013, pp. 300–308.

[165] Robert Ighodaro Ogie, "Adopting incentive mechanisms for large-scale participation in mobile crowdsensing: from literature review to a conceptual framework," *Human-centric Computing and Information Sciences*, vol. 6, no. 1, pp. 24, 2016.

[166] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp.

81–93, 2014.

[167] Prajakta Joglekar and Vrushali Kulkarni, "Mobile crowd sensing for urban computing," .

[168] Xinlei Chen, Xiangxiang Xu, Xinyu Liu, Hae Young Noh, Lin Zhang, and Pei Zhang, "Hap: Fine-grained dynamic air pollution map reconstruction by hybrid adaptive particle filter," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM, 2016, pp. 336–337.

[169] Osamu Masutani, "A sensing coverage analysis of a route control method for vehicular crowd sensing," in *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*. IEEE, 2015, pp. 396–401.

[170] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang, "Incentive mechanisms for crowdsensing: crowdsourcing with smartphones," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1732–1744, 2016.

[171] Dong Zhao, Xiang-Yang Li, and Huadong Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 1213–1221.

[172] Lingjie Duan, Takeshi Kubo, Kohei Sugiyama, Jianwei Huang, Teruyuki Hasegawa, and Jean Walrand, "Incentive mechanisms for smartphone collaboration in data acquisition and distributed computing," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1701–1709.

[173] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, 2012, pp. 173–184.

[174] Xinglin Zhang, Zheng Yang, Wei Sun, Yunhao Liu, Shaohua Tang, Kai Xing, and Xufei Mao, "Incentives for mobile crowd sensing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 54–67, 2016.

[175] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter, *Markov chain Monte Carlo in practice*, CRC press, 1995.

[176] Siyu Chen, Yong Li, Wenyu Ren, Depeng Jin, and Pan Hui, "Location prediction

for large scale urban vehicular mobility," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*. IEEE, 2013, pp. 1733–1737.

[177] Abhinav Jauhri, Brian Foo, Jerome Berclaz, Chih Chi Hu, Radek Grzeszczuk, Vasu Parameswaran, and John Paul Shen, "Space-time graph modeling of ride requests based on real-world data," *arXiv preprint arXiv:1701.06635*, 2017.

[178] "Openstreetmap. 2016. openstreetmap contributors. (2016) planet dump.," `http://http://wiki.openstreetmap.org/wiki/Main_Page`.

[179] "Beijing taxi data from computational sensing lab tsinghua university, 2016," `http://sensor.ee.tsinghua.edu.cn/datasets.html`.

[180] Yu Zheng, Xing Xie, and Wei Ying Ma, "Geolife: A collaborative social networking service among user, location and trajectory," *Bulletin of the Technical Committee on Data Engineering*, vol. 33, no. 2, pp. 32–39, 2010.

[181] Lili Cao and John Krumm, "From gps traces to a routable road map," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, New York, NY, USA, 2009, GIS '09, pp. 3–12, ACM.

[182] Yihua Chen and John Krumm, "Probabilistic modeling of traffic lanes from gps traces," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, New York, NY, USA, 2010, GIS '10, pp. 81–88, ACM.

[183] Marco Veloso, Santi Phithakkitnukoon, and Carlos Bento, "Urban mobility study using taxi traces," in *International Workshop on Trajectory Data Mining and Analysis*, 2011, pp. 23–30.

[184] L. I. Xiaolong, Gang Pan, W. U. Zhaohui, Q. I. Guande, L. I. Shijian, Daqing Zhang, Wangsheng Zhang, and Zonghui Wang, "Prediction of urban human mobility using large-scale taxi traces and its applications," *Frontiers of Computer Science*, vol. 6, no. 1, pp. 111–121, 2012.

[185] B. Jiang, J. Yin, and S. Zhao, "Characterizing the human mobility pattern in a large street network.," *Physical Review E Statistical Nonlinear & Soft Matter Physics*, vol. 80, no. 1, pp. 1711–1715, 2009.

[186] Genlang Chen, Xiaogang Jin, and Jiangang Yang, "Study on spatial and temporal mobility pattern of urban taxi services," in *Intelligent Systems and Knowledge Engineering (ISKE), 2010 International Conference on*, 2010, pp. 422 – 425.

[187] Xidong Pi and Zhen Sean Qian, "A stochastic optimal control approach for real-time traffic routing considering demand uncertainties and travelers' choice heterogeneity," *Transportation Research Part B: Methodological*, vol. 104, pp. 710–732, 2017.

[188] G. Pan, G. Qi, Z. Wu, and D. Zhang, "Land-use classification using taxi gps traces," *Intelligent Transportation Systems IEEE Transactions on*, vol. 14, no. 1, pp. 113–123, 2013.

[189] Guande Qi, Xiaolong Li, Shijian Li, and Gang Pan, "Measuring social functions of city regions from large-scale taxi behaviors," in *IEEE International Conference on Pervasive Computing and Communications Workshops*, 2011, pp. 384–388.

[190] Jing Yuan, Yu Zheng, and Xing Xie, "Discovering regions of different functions in a city using human mobility and pois," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 186–194.

[191] Nan Li and Guanling Chen, "Analysis of a location-based social network," in *IEEE Cse'09, IEEE International Conference on Computational Science and Engineering, August 29-31, 2009, Vancouver, Bc, Canada*, 2009, pp. 263–270.

[192] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[193] Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio, "Artificial neural networks applied to taxi destination prediction," in *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge - Volume 1526*, Aachen, Germany, Germany, 2015, ECMLPKDDDC'15, pp. 40–51, CEUR-WS.org.

[194] Fang He and Zuo Jun Max Shen, "Modeling taxi services with smartphone-based e-hailing applications," *Transportation Research Part C*, vol. 58, pp. 93–106, 2015.

[195] Michal Maciejewski, Joschka Bischoff, and Nagel Kai, "An assignment-based approach to efficient real-time city-scale taxi dispatching," *IEEE Intelligent Systems*, vol. 31, no. 1, pp. 68–77, 2016.

[196] Shuo Ma, Yu Zheng, and Ouri Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *IEEE International Conference on Data Engineering*, 2013, pp. 410–421.

[197] Haoyi Xiong, Daqing Zhang, Guanling Chen, and Leye Wang, "icrowd : Near-optimal task allocation for piggyback crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2010–2022, 2016.

[198] Yohan Chon, Nicholas D. Lane, Yunjong Kim, Feng Zhao, and Hojung Cha, "Understanding the coverage and scalability of place-centric crowdsensing," pp. 3–12, 2013.

[199] Asaad Ahmed, Keiichi Yasumoto, Yukiko Yamauchi, and Minoru Ito, "Distance and time based node selection for probabilistic coverage in people-centric sensing," in *Sensor, Mesh and Ad Hoc Communications and Networks*, 2011, pp. 134–142.

[200] H. Xiong, D. Zhang, G. Chen, and L. Wang, "Crowdtasker: Maximizing coverage quality in piggyback crowdsensing under budget constraint," in *IEEE International Conference on Pervasive Computing and Communications*, 2015, pp. 55–62.

[201] Juong-Sik Lee and Baik Hoh, "Sell your experiences: a market mechanism based incentive for participatory sensing," in *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*. IEEE, 2010, pp. 60–68.

[202] Georgios Amanatidis, Georgios Birmpas, and Evangelos Markakis, *Coverage, Matching, and Beyond: New Results on Budgeted Mechanism Design*, Springer Berlin Heidelberg, 2016.

[203] Zhenzhe Zheng, Fan Wu, Xiaofeng Gao, Hongzi Zhu, Guihai Chen, and Shaojie Tang, "A budget feasible incentive mechanism for weighted coverage maximization in mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[204] Luis G. Jaimes, Idalides J. Vergara-Laurens, and Andrew Raij, "A survey

of incentive techniques for mobile crowd sensing," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 370–380, 2015.

[205] T. Luo, S. S. Kanhere, J. Huang, S. K. Das, and F. Wu, "Sustainable incentives for mobile crowdsensing: Auctions, lotteries, and trust and reputation systems," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 68–74, March 2017.