Doctoral Thesis

Study on Crowd Sensing for Analyzing Human Activities and Urban Environment

(人間活動と都市環境分析のためのクラウドセンシングの研究）

Guangwen Liu

劉　　広文

Abstract

Pervasive smartphones that embed a variety of sensors enable us to sense and learn about not only the physical environment around us but also the society we live in. On the other hand, crowd sensing is a new sensing approach that individuals with sensing and computing devices collectively share data sensed or generated from their mobile devices, and aggregates the data in the cloud for discovering knowledge and solving problems. We designed a sensing platform based on crowd sensing paradigm to inexpensively acquire sensor data from smartphones, such as ambient noise, light, location, acceleration, temperature, etc., at a large scale. Based on the sensor data collected in our sensing experiments, we mainly studied trajectory computation, sensor data management, sensor application, and data analysis for specific purpose.

Trajectory simplification can greatly improve the efficiency of data analysis (e.g. querying, clustering). We applied information content theory in our method to simplify trajectory online. Moreover, we define a new error metric - enclosed area metric - to evaluate the accuracy of simplified trajectories, which is proven more robust against the uncertainty of GPS. Through comparing with other methods in a series of experiments over huge dataset, our method is proven effective and efficient.

The sheer volume of data collected through crowd sensing can deeply hamper the performance of various applications. We proposed a method to reduce the volume of sensor data while preserving the information content of the original data. That is, after data reduction, output data (target) can represent the original data (source). This method can compress multi-dimensional data without any assumption on distribution. We evaluated our method using real-world datasets. The results show that our method outperforms one state-

of-the-art and two other conventional baseline methods based on data divergence.

Accurate estimation of elevation is important for many location based services. It is possible to use barometers on smart phones to estimate elevation in both indoor and outdoor scenarios. However, to reach an acceptable level of accuracy, a reference point which periodically broadcasts its air pressure and temperature is required. We proposed a method to increase the spatio-temporal density of reference points by exploiting neighboring smart phones as ad-hoc reference points. In addition, we also employed Kalman filter to stabilize the elevation profile due to the instability from smart phone sensors. Experiments conducted in both indoor and outdoor with different geographical characteristics reveal that our system can provide elevation with an error less than 5 meters in 90 % cases and less than 3 meters in 75 % cases, which is sufficient for most practical applications.

In addition, three data analysis cases on the collected data are studied independently: estimating nighttime activity, building noise maps and sensing micro-weather, which fully leverage the power of crowd sensing to find collective intelligence.

Overall, in this thesis, we studied how to leverage crowd sensing to analyze human activities and urban environment, including sensing platform design, sensor data management, special sensor application, several data analysis cases. Since crowd sensing is still at its infancy, there is still a lot of work left for us, such as robust frame work, privacy preservation, incentive mechanism, and large scale useful applications.

Acknowledgements

I could never have completed this work without the support and encouragement from many people. First and foremost, I would like to express deep gratitude to my advisor Prof. Dr. Sezaki Kaoru for his valuable guidance and generous support. With his help, I quickly converted myself to a researcher from an engineer. It is very fortunate that I could do state-of-the-art research in such a liberal and encouraging atmosphere which Prof. Sezaki fostered in the laboratory. He provided me with numerous advice and insightful comments not only on academic research but also on personal daily-life and future career, which will have a profound effect on my life.

I also would like to express grateful thank to Assoc. Prof. Iwai Masayuki. He always encourage me to challenge higher achievements. He closely worked along with me in my research project even after he is promoted to Assoc. Prof. in Tokyo Denki Univ. from Univ. of Tokyo. He also provided me good opportunities to do joint research work with industrial company. Besides, I would like to express my special appreciation to Professor Tobe, he gave me extremely helpful suggestions on research in his busy time.

I am grateful to work with my lab-mates, especially Dr. Asif who often discusses with me and spares no effort to help me on my research and personal life, Dunstan who also helps me so much with paper writing. The collaboration with them made me stay keen on my research. Assistant Prof. Kobayashi and Assistant Prof. Ito also provided me with sincere supports. I truly acknowledge their kindness. I also appreciate the help from all other

seniors, juniors and secretaries in our laboratory.

Finally, I wish to express special thanks to my family. My parents and young brother always stood by me and supported me throughout my entire life. I would like to give my deep gratitude to my wife Lyli whose patient love enables me to concentrate on the research and complete my doctoral course.

August 2014

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Human beings walked from feudal society to capital society, strode from absolutism to democratism. The most revolutionary change is that the common people or the majority of people are considered as the primary source of political power. I don't plan to discuss or demonstrate the advantage of specific political ideology here, however, i would like to refer to James Surowiecki's book, *"The wisdom of crowds"*, in which it states under the right circumstances, groups are remarkably intelligent, and are often smarter than the smartest people in them [Sur05]. Groups do not necessarily need to be dominated by super-intelligent people in order to be smart. Even though most of the people within a group are not really well-informed, it can still reach a collectively wise decision. In fact the solutions to coordination, cognition and cooperation problems by collective wisdom are at work in real situations in society every day.

There are four necessary conditions for a crowd to be wise according to the book: diversity of opinion, independence, decentralization and aggregation. If a group meets these conditions, its output is likely to be accurate. In my opinion, the reason is easy to be proven by mathematics. If you ask a large

enough group of diverse, independent people to make a prediction or estimate a probability, and then average those estimates. It is similar to estimate the expected value by obtaining from a large number of trials. Therefore, it is consistent with *the law of large numbers*.

In the domain of information technology, the similar idea coined the term *"crowdsourcing"* is fully leveraged in many services too, including Wikipedia, Linux, Yahoo! Answers, Amazon Mechanical Turk. According to the definition by Merriam-Webster dictionary, *crowdsourcing* is the practice of obtaining needed services, ideas, or content by soliciting contributions from a large group of people and especially from the online community rather than from traditional employees or suppliers. These well known projects (e.g. Wikipedia, Linux) exactly practiced the idea of crowd sourcing, to which a large group of people contributed with their own wisdom.

Nowadays mobile phones or smartphones are becoming ubiquitous and they don't only serve as computing and communication devices but also comes with a rich set of sensors built-in, such as GPS, accelerometer, microphone, gyroscope, camera, blue tooth, and so on. It draws attention from many researchers, and forms a new research domain called mobile sensing [LMLe10, KXAA13]. Thus, with the consideration of the participation of large number of mobile phone users, researchers come up with a new field termed *mobile crowdsensing* [GYL11]. Raghu Ganti [GYL11] defines *mobile crowdsensing* as individuals with sensing and computing devices collectively share data and extract information to measure and map phenomena of common interest. Other researchers also formally define it as a new sensing paradigm that empowers ordinary citizens to contribute data sensed or generated from their mobile devices, aggregates and fuses the data in the cloud for crowd intelligence extraction and people-centric service delivery [GYZZ14].

In [GYL11], mobile crowdsensing is ascribed to a subcategory of Internet of Things (IoT). Typical devices of IoT are RFID (Radio-Frequency IDentification) tags, sensors, mobile phones, which have unique id, sensing capability

and can communicate with internet [AIM10]. Mobile crowdsensing meets all the requirements of IoT, consequently it can be seen as a subcategory or even an evolution of IoT. Unquestionably, IoT will revolutionize our society and economy because it has high impact on several aspects of our everyday life and behavior of potential users. Therefore, recently mobile crowdsensing is becoming a hot research area.

Before the advent of mobile crowdsensing, there is also a nearly same concept called participatory sensing. Today billions of people carry mobile phones and these ubiquitous devices are increasingly capable of sensing ambient environment and human activities. Under such a condition, Burke et al. [BEHea06] proposed the idea of participatory sensing which task deployed mobile devices to form interactive, participatory sensor networks that enable public and professional users to gather, analyze and share local knowledge. Although the definition and the background of *participatory sensing* and *mobile crowdsensing* are extremely close, there are some differences. Mobile crowdsensing consists of personal sensing and community sensing while participatory sensing only refers to community sensing. In addition, participatory sensing emphasizes explicit user participation while crowdsensing involves both implicit and explicit participation. Moreover, crowdsensing collects data from two data sources: sensor data and user-participant social data. According to the definition by [GYZZ14], mobile crowd sensing unified the human intelligence (from participatory sensing) with machine intelligence (from crowdsourcing).

Since most applications of this field deployed in urban environment (e.g. participants are citizen usually, sensing activities are basically conducted in urban, and most applications aims to solve urban problems including traffic, pollution and mobility), researchers also call it *urban sensing* [CELea06, CEL+08, LEM+08]. In my point of view, urban sensing is not comprehensive to define this new sensing approach, thus I use the concept of crowd sensing for my work.

Although crowdsensing is born recently, there are many fruitful research work in the literature. These work can be roughly classified to five categories: sensor data analysis applications, sensing architecture design, privacy preservation, participant incentives, energy-efficient sensing.

- Sensor data analysis applications

  By leveraging sensors built in mobile phones and participants themselves, many interesting applications are developed to solve urban problems or discover knowledge. The work in [CG13] aims to easing a typical traffic issue - helping drivers easily find vacant parking spaces. It focuses on identifying legal parking spaces from crowdsourced data. A research conducted by Microsoft Research also targets at the problem of monitoring road and traffic conditions in a city by using smartphone[MPR08]. They present a system that performs rich sensing by piggybacking on smartphones, which uses the accelerometer, microphone, GSM radio, and/or GPS sensors in these phones to detect potholes, bumps, braking, and honking. Another work by Eiman Kanjo [Kan10] presents a real-time mobile phone platform for urban noise monitoring and mapping. Cycling couriers carry Nokia mobile phones to collect noise data around Cambridge city. In [MSNS09], a similar work to monitor noise level by normal citizen has been done. In the work [CLLea12, CKC13] it exploits opportunistically captured images and audio clips from smartphones to link place visits with place categories (e.g., store, restaurant). The system combines signals based on location and user trajectories along with various visual and audio place "hints" mined from opportunistic sensor data. Their result shows it can classify places into a variety of categories with an overall accuracy of 69%.

  Aside from sensor data exploring, there are other interesting work which exploit participant users' intelligence. A significant work called CrowdSearch combines automated image search with real-time human validation of search results. Automated image search is performed using a combination of local processing on mobile phones and back-end processing on

remote servers. Human validation is performed using Amazon Mechanical Turk, where tens of thousands of people are actively working on simple tasks for monetary rewards [YKG10]. Another work called CrowdDB [FKK+11] also applied the similar idea - using crowd's wisdom. It uses human input via crowdsourcing to process queries (SQL) that neither database systems nor search engines can adequately answer. Besides, the work in [DL14] presents general algorithms for efficient human meta-data collection, which can be used to label and/or validate the primary sensor data.

- Sensing architecture design

  Researchers also designed several general frameworks for crowd sensing system from different viewpoints. Medusa is a general programing framework for crowd sensing applications which support for humans to trigger sensing actions or review results, the need for incentives, as well as privacy and security. Medusa provides high-level abstractions for specifying the steps required to complete a crowdsensing task, and employs a distributed runtime system that coordinates the execution of these tasks between smartphones and a cluster on the cloud [RLLPG12]. The work in [JSS+12] shows a basic architecture which consists of data collection & analysis module and context aware data processing module. In the work [SJK+12] a platform is developed to support data collection for mobile crowdsensing, which results in reducing the amount of data sent, as well as the energy usage on the mobile phone, while providing comparable levels of accuracy to traditional models of intermittent/continuous sensing and sending. This paper [ZJK13] also focuses on data collection and mining approach. It presents an effective approach to address the scalability issues of data collection and run-time processing. These work [HLZ+13, TCS+12, CFB+13] try to perform crowd sensing task in an efficient manner. They focus on how to maximize conditions for user participation. Vita [HCCL13] provides a flexible and universal architecture across mobile devices and cloud

computing platforms by integrating the service-oriented architecture with resource optimization mechanism for crowdsensing. This work [KHKZ08] also considers to offer optimal sensing policies under the constraints of resource and privacy limit. Yohan Chon et al. [CLK+13] study on how to deploy successful crowd sensing systems in terms of coverage and scalability. They deployed a large scale place-centric crowdsensing system to examine place-temporal coverage, relationship between user and coverage and privacy concerns.

- Privacy preservation

  A main obstacle to crowdsensing's widespread deployment is the privacy concerns of participating individuals. Consequently, some research works focus on the issue about how to preserve privacy in sensing activities. The work in [PXGUS14] assess the threats to participant privacy when personal information is disclosed. It outlines how privacy mechanisms are utilized in existing sensing applications to address these threats. AnonySense [SCP+11] allows applications to submit sensing tasks to be distributed across participating mobile devices, later receiving verified, yet anonymized, sensor data reports back from the field, thus providing the first secure implementation of this participatory sensing model. PriSense [SZL10] is a novel solution to privacy preserving data aggregation in crowd sensing systems. PriSense can support strong user privacy against a tunable threshold number of malicious users and aggregation servers.

- Participant incentives

  For most crowd sensing system, a major issue is how to recruit participants as many as possible. To address this issue, the work in[YXFT12] designs incentive mechanisms for crowd sensing. They consider two models: the platform-centric model where the platform provides a reward shared by participating users, and the user-centric model where users have more control over the payment they will receive. For the platform-centric model, an

incentive mechanism using a Stackelberg game is introduced to maximize the utility of the platform. For the user-centric model, an auction-based incentive mechanism is introduced to keep being fair and profitable. In this work [TBH13] it proposes a framework with two main components: participant recruitment and data collection, which adopts a new approach to match mobility profiles of users to the coverage of the sensing mission. However, its result is only evaluated by extensive trace-based simulations. On the other hand, this work [RES10] develops a recruitment framework to enable organizers to identify well-suited participants for data collections based on geographic and temporal availability as well as participation habits. And it is proven effective in coverage-based recruitment through a real sensing campaign.

- Energy-efficient sensing
  Since participants are volunteers or paid with limited reward, imposing heavy energy burden on their mobile phones will discourage the participation motivation. Therefore, there are many work to mitigate this energy issue. Nicholas Lane et al. [LCZ+13] propose a system for collecting mobile sensor data from smartphones that lowers the energy overhead of user participation. Their approach is to collect data when smartphone users place phone calls or use applications. In these situations, the energy needed to sense is lowered because the phone need no longer be woken from an idle sleep state just to collect data. It builds a prediction model of application usage to drive a decision engine that lets the smartphone locally decide which application opportunities to exploit based on expected energy/quality trade-offs. This work [HZY] proposes a distributed algorithm for maximizing the utility of sensing data collection when the smartphone cost is constrained. It applies stochastic network optimization technique and distributed correlated scheduling to determine the optimum sensing utility. The work in [WZX13] reduces the data cost of non-data-plan users by maximally offloading the data to Bluetooth/WiFi gateways while it re-

duces energy consumption of data-plan users by uploading data in parallel with a call or using less-energy demand networks (e.g. Bluetooth).

1.2 Motivation

Thanks to the development of sensor technology and the pervasive usage of smart phone, it enables us to sense and learn our physical environment even the social activities. If billions of smart phones, which are deployed in all over the world, sense whatever happened on the earth and store it in the cyberspace, what will the world be? We can create a digital earth in cyberspace which fully maps or records what occurred in the past and what occurs now on the real world. In contrast to Google Earth, this will be a dynamic digital earth which is enriched with real-time physical phenomenon and human social activities. Undoubtedly crowd sensing is one of the most effective approaches to build such a global level cyber-physical system.

Although crowd sensing is developing rapidly in the recent years, there are countless issues left to be solved for us. Compared to traditional sensing approach - wireless sensor networks (WSN) [ASSC02], crowdsensing has unique characteristics.

- First, modern mobile phones posses much more computing power, communication and storage resource than sensors in WSN, and they are equipped with multi-modality sensing capabilities (e.g. there are about 16 types of sensor on modern smartphone). In future it is expected to have more powerful mobile devices, such as smart glass, smart watch, smart clothes, and so forth. This strength enables more powerful and more extensive applications.
- Second, billions of mobile phones are already deployed in the field: users carry their phones anytime and anywhere. It is possible to build large-scale sensing platforms with less cost and less time by making

advantage of this characteristic. But in real systems, to recruit users for participation becomes a central issue.

- Third, because it always involve with people, thus we can sense both our environment and social activities. For example, GPS, blue-tooth and/or opened applications history can be used to infer user's social activities. On the other hand, it brings about new problems - privacy protection and incentive measures.
- Fourth, sensors are mobile, thus sensing context is dynamic. Consequently data quality is not easy to control and we need to design new architectures to organize the entire system. In addition, it is particularly challenging to fuse information in a heterogeneous network which integrates static sensor infrastructure with mobile sensors.

These unique characteristics open a new door to solve complicated problems, at the same time it brings about new challenges. Nevertheless many researchers are motivated to work on it. In the literature, numerous work has been done on novel applications, architecture design and effective methodology for energy saving, privacy preservation and participant recruitment. However crowd sensing is still in its infancy, thus it motivates me to study further.

More specifically, I mainly concerned on the following problems:

- How to utilize smartphone sensors and manage sensing activities?
  Smartphone is not specifically manufactured for sensing. To utilize sensors and to facilitate sensing, it is necessary to develop a particular software to provide sensing service and manage sensing activities for participants.
- How to handle huge volume of sensor data?
  Crowd sensing can easily acquire big data. However, smartphones are not dedicated to sensing, thus heavy load on transmission and power

consumption will hinder people to participate. In addition, huge volume of sensor data also will hamper the efficiency of data analysis on the collected data.

- How to acquire highly accurate sensor data?
  Sensors built-in smartphones is not as good as the dedicated sensors. Although they are cheap and widely exist, if the accuracy is not guaranteed, the sensor application and sensor data analysis result are no meaningful or less reliable.

1.3 Contributions

The main contributions of this thesis can be summarized as follows:

- We designed a general sensing platform by applying crowd sensing paradigm and developed a flexible sensing tool on smart phone Then a series of large scale sensing experiments based on this platform were conducted to collect data.
- We defined a new metric to measure trajectory similarity which can be used in trajectory computation (e.g. trajectory simplification, search) for mobility analysis.
- To handle huge sensor data computation, we presented a novel method to manage sensor data, which can efficiently reduce the volume of data set while preserving the value of data.
- By leveraging sensor data collected in our experiments, we discovered interesting knowledge in three scenarios: estimating nighttime activity, building noise maps, sensing micro-scale weather.
- In addition, we developed a novel application which can accurately measure elevation by using smartphone's barometer.

The rest of the thesis is consisted of 6 chapters: In chapter 2, the overview of sensing platform is outlined and our developed sensing tool is introduced.

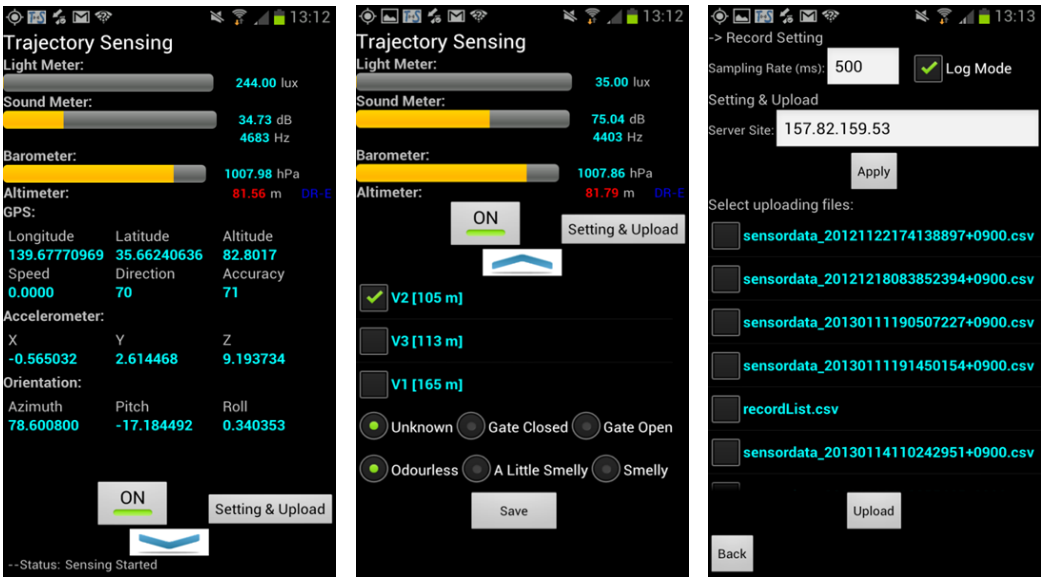
Then, sensing experiments conducted in Setagaya-Ku are stated and a particular sensing experiments in Tianjin, China is also briefly explained. In chapter 3, it introduces a new similarity metric based on enclosed area to measure the displacement between two trajectories. Through simulation and theoretical analysis, this new metric is proven more robust against GPS uncertainty comparing to traditional metrics. Based on this new metric, a trajectory simplification method is proposed, which is evaluated in terms of accuracy on large real dataset. In chapter 4, it defines a new data model called *REPSense* to represent data from the viewpoint of data diversity. Then describes how to implement this model with a divide/merge scheme. Based on large real dataset, a series of experiments are conducted to evaluate the performance of our method in terms of data divergence and data clustering performance. In chapter 5, an integrated framework to provide accurate elevation measurement is proposed. It employs multiple techniques to handle systematic error and random error, integrates heterogeneous reference points, applies Kalman filter to smoothen elevation profile, and leverages short-term forecast model to enhance the accuracy. The system is well evaluated in different indoor and outdoor settings, i.e. outdoor walking, mountain climbing, inside buildings. In chapter 6, three data analysis cases are studied independently. In section 6.1, it describes how to estimate nighttime activity from ambient noise and ambient light. In section 6.2, it reports how a noise map for residential area is built by using crowd sensing. In section 6.3, it introduces weather sensing that acquire fine-granularity weather factors and fuse information from both mobile and static sensors. In chapter 7, conclusions of the thesis are drawn, some future directions are presented.

Chapter 2

Sensing Platform

Mobile phone has evolved into smart phone. Nowadays smart phone can be used not only for communication but also entertainment, computation and sensing. With significant improvements in mobile sensor technology, smart phone is becoming smarter and smarter. Smartphone comes with a growing number of powerful embedded sensors, such as an accelerometer, magnetometer, gyroscope, GPS, microphone, and camera. The figure 2.1 shows 16 kinds of sensors built in a modern smartphone (Samsung S4). Therefore, smartphone knows where you are, what you say, what you see, how fast you are moving, and physical environment including ambient noise, illumina-

Figure 2.1: Sensors built-in smartphone



(a) (b) (c)

Figure 2.2: Sensing tool on smartphone

tion, temperature, humidity and air pressure. Yet this is only the beginning chapter in the era of context-aware devices.

To explore all kinds of sensors, we developed a sensing tool on *Android* smartphone (see figure 2.2), called *trajectory sensing tool*. This tool logs all kinds of sensor data, including light, sound, air pressure, temperature and humidity (only available in some smartphones), GPS (latitude, longitude, altitude), acceleration, orientation, proximity and so on. It is implemented as back-end services, thus it can run in background without disturbing user's other operation. Basically it is assumed to record the data along with trajectory but you can set it to measure only at specific locations. It also provides uploading data file to server for sharing. In addition, its core services is encapsulated to a library, consequently it can be built into the third party software.

Figure 2.3 shows the architecture of this tool, which consists of 7 components. The *sensing service* calls *sensor manager* of Android OS to start

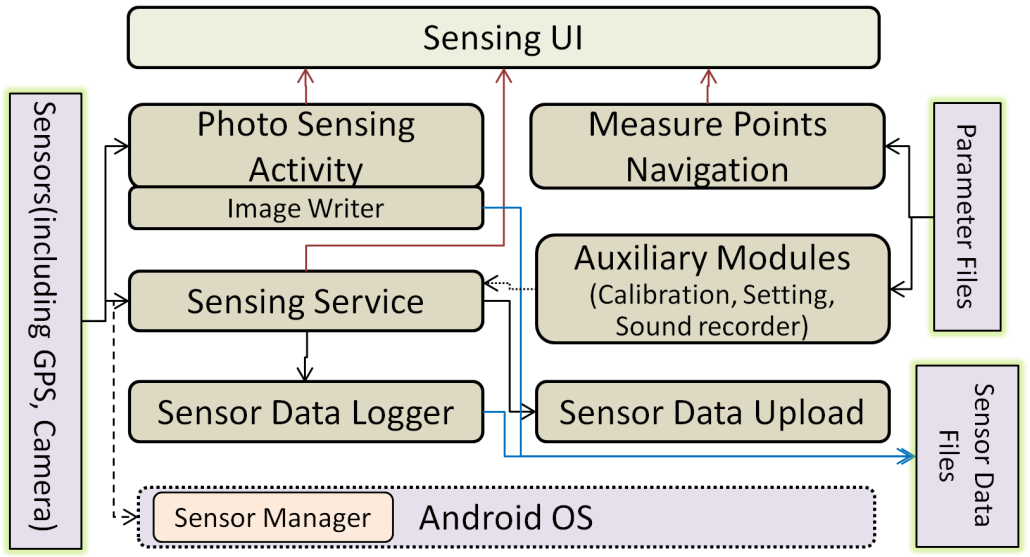


Figure 2.3: The architecture of the sensing tool

sensors, the sensor data will be logged into files by *Sensor Data Logger* and be transmitted to the backend server by *Sensor Data Upload*. *Auxiliary Modules* provides calibration for sensors, setting for sensing parameters (such as sampling rate, sensor selection), and particular service for microphone (Sound recorder). *Photo Sensing Activity* can take photos with location and orientation, and synchronize other sensor data. *Measure Points Navigation* helps in traversing all measurement points for fixed point sensing. *Sensing UI* provides an interface for participant to operate sensing activities and visualize sensor values.

According to crowd sensing paradigm, we designed our sensing platform as shown in figure 2.4. Participant users carry their smartphone with our developed sensing tool to collect data during their daily-life activities, e.g., on the commuting route. The tool will make some preliminary process, e.g., calibrating sensor values and removing redundant data, before uploading to back-end server. In addition, at the side of mobile phone, particular sensors are involved to make special applications, e.g. measure elevation from air pressure sensor, and calculate sound level from microphone. At the side

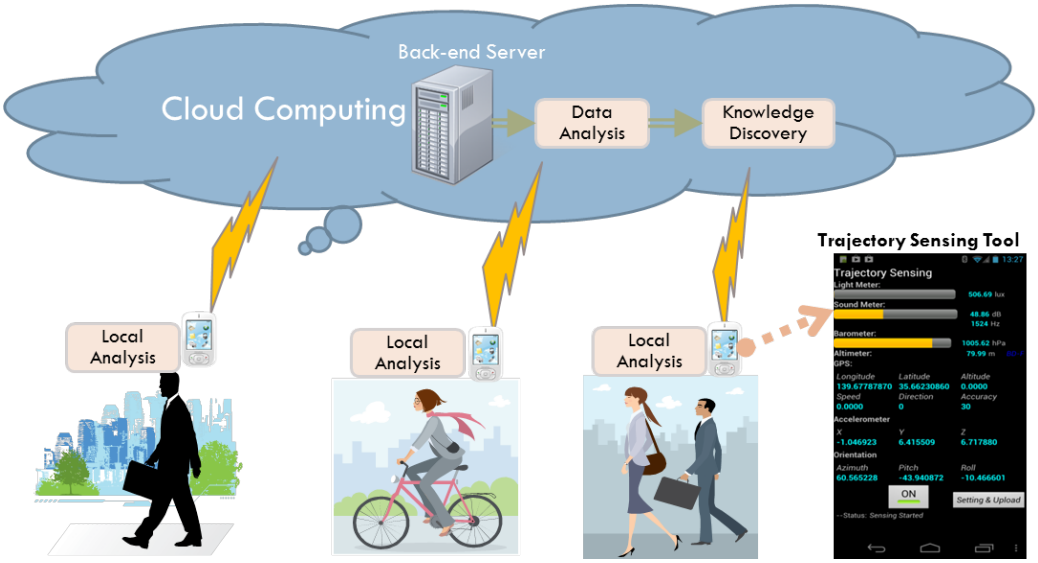


Figure 2.4: Sensing platform overview

Figure 2.5: Walking with sensing tool

Figure 2.6: All sensing traces in Setagaya-Ku, Tokyo

of server, sensor data is aggregated from a number of participants, then we make comprehensive data analysis to find collective intelligence. For instance, in our studies, we build noise map, analyze nighttime activity, and predict micro-level weather.

Based on this sensing platform, we conducted a large scale experiment in Setagaya-Ku, Tokyo. We employ different models of smartphones (including Samsung S3, S4 and LG Nexus 4) for experiments. Sampling rate is set to 2 Hz or 50 Hz in different cases. About 40 people participated this experiment, and it took 5 days to walk through this area (about 60 square kilo-meters). Participant bind the smartphone with armband on their arm (see figure 2.5) or hold it at hand or put it in pocket but keep light sensor visible from outside. Since illuminance measured by light sensor will be meaningless if the smartphone is hidden inside the pocket, we required the participants to use armband or hold it at hand. However, due to the human error, participants still blocked the light sensor. Fortunately, there is a prox-

Figure 2.7: Photo sensing in old foreign settlement, Tianjin

imity sensor which can determine whether the smartphone (light sensor) is blocked or not. Figure 2.6 shows all the traces that participants walked.

In addition, by leveraging this tool, we conducted an experiment in Tianjin, China. In this experiment, the purpose is to investigate the old buildings in foreign settlement of Tianjin for heritage conservation. Participants take photos when walking around the buildings. When taking photos, the position (including horizontal and vertical position) and orientation are also recorded, thus photos can be mapped to the map with actual view angle. Figure 2.7 shows an example of photos took in this experiment. In addition, photos are stored along with other sensor data which depicts ambient environment, for example, ambient noise, light, temperature, humidity. The photos and

sensor data can be used to analyze the condition of buildings and peripheral situation. However, to report the detail of this project is outside of my range.

Chapter 3

Sensing Trajectory Simplification

3.1 Introduction

Nowadays, GPS-enabled devices, ranging from smart phones to vehicles, are drastically increasing. Moreover, Location-based services and applications built from GPS-equipped mobile devices is a rapidly expanding consumer market, e.g., fleet management, traffic analysis and scientific investigations[Kup05]. In addition, recently there has been a promising application called mobile crowd sensing or participatory sensing by smartphones[GYL11, LMLe10]. We can collect many rich and useful data, e.g. noise, illumination and temperature, just by normal users who travel with smartphone in daily life. These sensor data is usually generated along with trajectory. Although data generated from GPS devices are commonly used in a variety of businesses, these efforts will be hindered by the massive volumes of data, which creates the problem of storing, transmitting, and processing[LRH11]. Storing the data is difficult because the sheer volume of data can rapidly overwhelm available data storage. For instance, if data is collected at 30 seconds intervals for 400 users with GPS-equipped smartphone, the volume will be up to

1.1 GB in a month. In addition, these trajectories will cause a heavy load for network transferring which costs highly in view of money and time. The cost of sending large volume of data over remote networks can be prohibitively expensive, normally ranging from 5 to 7 per megabyte[Jon08]. The foremost issue is that the enormous volume of data can easily overwhelm human analysis and further computing. For example, towards querying and clustering trajectories, the performance will exponentially decrease due to the number of position data[YAS03][WSKe11].

The above restrictions motivate the need to reduce data volume. Moreover, trajectory data is usually collected in a random manner; consequently a part of information is redundant and reducible. Hence, numerous compression methods have been proposed to reduce the size of trajectory data sets[PPS06][MPP11]. However, these methods often either lose some contextual information or are computation-expensive. Besides, conventional compression methods, such as LZ (used in zip) or DCT (Discrete Cosine Transformation) can compress the data volume, whereas it does not improve the data processing efficiency (e.g. querying, clustering) as data should be uncompressed to original volume before processing. In this chapter, we present a novel compression method to quickly simplify the trajectory before the position data is transmitted to the server from GPS terminals. Our contributions can be summarized as follows:

- We propose a simplification method based on MBR of information content, which can largely keep as same information content as counterpart of original trajectory.
- We also introduce a new error metric based on enclosed area to measure the displacement between original trajectory and simplified one.
- Through simulation, enclosed area metric is proven more robust against GPS uncertainty comparing to distance metric.
- Evaluate the accuracy with other typical simplification methods in

terms of perpendicular distance, synchronized Euclidean distance and enclosed area. In addition, we estimate the effect of parameters over the performance of our method.

The next section describes related work about compressing trajectories. In section 3.3 our method is described in detail. The evaluation of our method and other algorithms with 3 error metrics including newly introduced metric is described in section 3.4. Finally, discuss experimental results and future work.

3.2 Related Work

In the literature various simplification methods exist [GKMea07][Tay05][CWT06] and Lawson et.al conducted a very comprehensive survey for them[LRH11]. Most widely used methods are uniform sampling, dead reckoning method and Douglas Peucker method, which also are the comparison targets in our work. We will introduce these existing methods and analyze their advantages and disadvantages here.

3.2.1 Uniform Sampling

Uniform sampling is a naive method which sparsely selects the point to store by every given time interval or distance interval but discards remained points. In some applications, this method is modified by storing the average value of all points within given interval, which is called piece-wise aggregate approximation. Even though uniform sampling may provide a simple and cost-effective solution, it is distinctively insensitive to the spatio-temporal characteristics of the trajectory as well as to its sequential nature. Hence, we can't expect the consistent quality just since it is too sensitive to the case, though the result is satisfactory in some case.

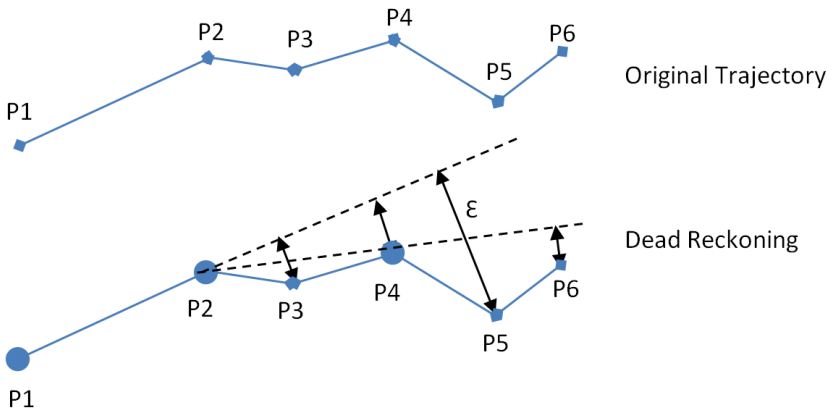


Figure 3.1: Dead reckoning simplification

3.2.2 Dead Reckoning Method

It is a localized processing routine which make use of the characteristics of the immediate neighbouring coordinate points in deciding whether to retain the current point. As shown in figure 3.1, P_3 and P_4 are in the same trend with the line segment consisted of P_1 and P_2. However, since P_5 exceeds the threshold of Euclidean distance predefined, the prior point of P_5 will be retained in the simplified trajectory so that the maximum distance displacement does not go beyond the predefined ϵ. This method has two advantages: (1) it can process the data at local client (mobile terminals) (2) its time complexity is $O(n)$, namely linear. Therefore, it is popular in car navigation though it accumulates the error in bad case. There are also some variants of this method [PPS06][LDR08].

3.2.3 Douglas Peucker Method

DP method was proposed by Douglas and Peucker [DP73], which is widely used in cartography related software like AUTOCAD. "Many cartographers consider it to be the most accurate simplification algorithm available, while others think that it is too slow and costly in terms of computer processing

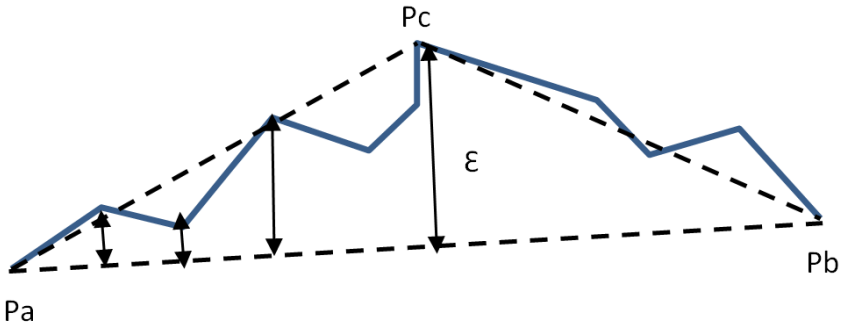


Figure 3.2: Douglas Peucker simplification

time" [Jen89]. In any case, it is the most famous line simplification algorithm till now. The method recursively selects two points to represent the line segment within a specified tolerance value (see figure 3.2). Firstly, it attempts to simplify the trajectory with $\overrightarrow{P_aP_b}$, but it discards this attempt when calculate perpendicular distance from every point to line $\overrightarrow{P_aP_b}$ and find P_c is out of the predefined threshold ϵ. Then, it chooses P_c as new anchor point and repeats the attempts with $\overrightarrow{P_aP_c}$ and $\overrightarrow{P_cP_b}$ respectively.

As described above, the algorithm of Douglas Peucker is simple and easy to program. It is extremely efficient because it globally exhibits the least distance error under a given tolerance. Thus, it is recognized as the one that delivers the best perceptual representations of the original lines [WM03]. Moreover, Douglas Peucker Method is applicable to both 2D line and 3D line in terms of any shape of line. Besides, as its basic idea is universal, it has been independently proposed in other contexts, i.e. image processing, computational geometry, and there is also many works that try to improve this method[HS92]. Nevertheless, this method loses its power when the trajectory includes self-intersection points. In addition, this method is very computing-expensive in some cases. A straightforward implementation requires O(n) time to find the furthest point from line. Since the iteration depth is linear, the worst-case running time is $O(n^2)$.

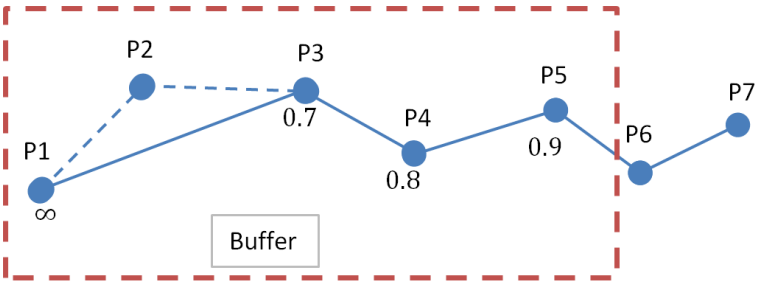


Figure 3.3: SQUISH simplification

3.2.4 Other Methods

Aside from these conventional methods, recently researchers also presented other interesting methods. Yukun Chen et.al [CJZe09] proposed such a method to simplify trajectory for LBS networking services. The method focuses on keeping speed and direction change information as much as possible. Hence, they defined the attributes of line segments, including heading direction, neighbor heading change, accumulated heading change, heading change which is the sum of the neighbor heading change and accumulate heading change, and neighbor distance. Then, the method assigns the weight on point in terms of the product of the average heading change and the neighbor distance. Lastly, the method selects the points with high weight to represent all the sampling points. Their approach is said to outperform Douglas-Peucker method in walking mode trajectories with some constraints. In any case, it is a global process routine so we will not compare it with ours since the purpose of our method is to process data online at the mobile terminal.

Besides, the STTrace algorithm[PPS06] is designed to preserve spatiotemporal heading and speed information in a trace. A hybrid between an online and offline approach, STTrace defines a safe area by using the previous two points in the trajectory. A vector which defines the speed and heading between the two locations is used to predict the position of the next point. It uses two input parameters to make this prediction. One of these parameters

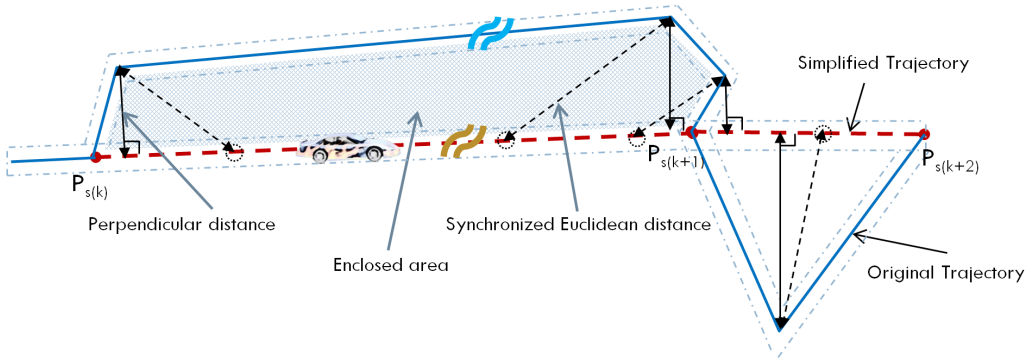


Figure 3.4: Error metrics

is the speed threshold which defines how much the speed can vary while still remaining in the predicted range. The other input parameter is the heading threshold that defines how much the heading can vary while still remaining in the predicted range. STTrace is similar to dead reckoning method while it uses the predicted area to filter not the fixed point as in DR method. Although it can overcome complications caused by error propagation, which improved the demerit of DR method, its processing time is far worse than the threshold-based methods.

In addition, Jonathan Muckell et.al [MPP11] proposed a method called SQUISH to simplify trajectory using a priority queue. As shown in figure 3.3, the method sets a buffer for processing points in which all points will be assigned a weight. The weight value is determined based on estimating the amount of synchronized Euclidean distance introduced into the compression if that point was removed from the trajectory. It is straightforward to delete the points with the lowest weight in order to keep the shape of trajectory as same as possible, whereas it is a little confusing to add the deleted point's weight on the neighbor point. For example, P_2 with the lowest weight is moved out from the buffer, but the weight of P_3 cannot be simply obtained by adding the weight of P_2. The experiment of this method demonstrates a good result comparing with other methods when the compression rate is not large, otherwise it lost the lead.

All these methods are data-loss methods, though there are data-lossless compression methods in other fields [BR07]. Usually data-lossless compression is computing-expensive, and for trajectory to some extent, data-loss is acceptable in most cases. Hence, the point is to diminish the data-loss under arbitrary compression ratio. That is, the data-loss measures or error metrics are also extremely important.

However, all existing methods try to approximate trajectory based on distance measure, including *perpendicular distance* and *synchronized Euclidean distance* (see figure 3.4, formal definitions are stated later). As it can be seen in figure 3.4, in some cases, e.g. between $P_{s(k)}$ and $P_{s(k+1)}$ there are two long parallel lines, even when the distance displacement is slight, the enclosed area displacement will be quite remarkable. Moreover, occasionally *GPS* position is quite inaccurate so that certain points have large displacement, and discrete metric is extremely sensitive to these contingent errors but continuous metric is more resistant to them. It motivates us to develop our method based on enclosed area. In the later section, we will fully prove that enclosed area is a more accurate error metric while considering GPS uncertainty.

3.3 Proposed Method

There is a wide variety of sensors built in smartphone, and *Android SDK* support tens of sensors including accelerometer, ambient temperature, gravity, gyroscope, light, magnetic field, orientation, pressure, proximity, relative humidity, etc. Aside from them, *GPS*, *Bluetooth* and *Wi-Fi* are also available. By making use of these sensors, we launched a project called *trajectory sensing* to sense the physical environment (including ambient noise, light via microphone and light sensor) and human activities (via accelerometer, orientation sensor).

3.3.1 Problem Statement

In our crowd sensing platform as stated in chapter 2, participants take smartphone installing our sensing tool during their commuting routes by walking, bike or car. Finally the data is transmitted to back-end server via cellular network. The ultimate goal of our project is to learn human activities under particular physical environment by sensing and then to discover knowledge. However, the problem we plan to solve in this work is trajectory simplification (i.e. data reduction) in real-time before data analysis.

Trajectory is obtained by recording the successive positions of which a moving object takes across time. Recently, researchers try to enrich trajectory (called *semantic trajectory*) by adding background geographic information to discover meaningful patterns[YLWea11]. To extend this concept further, we redefine *semantic trajectory* T as:

$$T = \{p_t = [X(t), Y(t), Z(t), Sm(t)] \mid t \in R\} \tag{3.1}$$

Here, p_t is a tuple from one data point recorded as time t, including spatial position $(X(t), Y(t), Z(t))$ and semantic attributes $(Sm(t))$. Spatial position usually refers to latitude, longitude and altitude, and semantic attributes refers to light, noise, temperature and so on. On the other hand, after data reduction a simplified trajectory T' can be defined as:

$$T' = \{p_t = [X(t), Y(t), Z(t), Sm(t))] \mid t \in R \cap \Delta(t) > \varepsilon)\} \tag{3.2}$$

In fact, we can describe the nature of data reduction problem as constructing a threshold function Δ(t) to achieve the least data loss within favourable or given compression ratio. Our method, just like the existing method is also expected to reach this goal.

After data reduction, the amount of data points decrease but data value and the order are never changed, that is, reduced data meets equation (3.3)

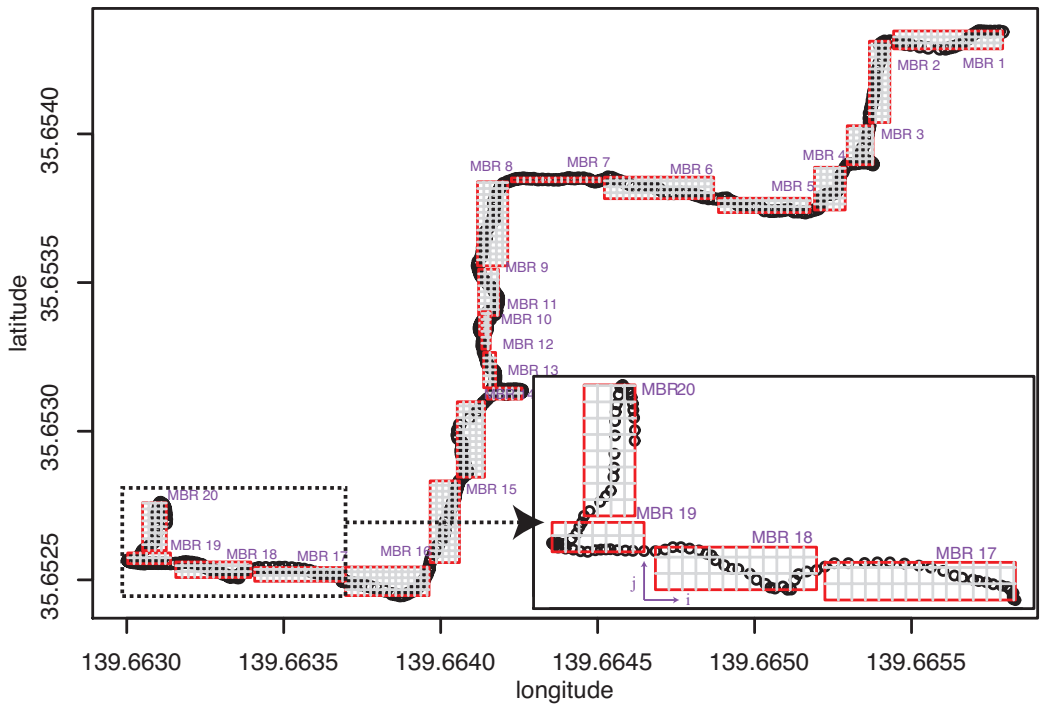


Figure 3.5: A sample data of trajectory with MBR

and (3.4). We call these two properties *value-invariance* and *order-invariance* respectively.

$$\forall p' \in T' \;\Rightarrow\; \exists! p \in T : p' = p \tag{3.3}$$

$$\begin{gathered}\forall p', q' \in T' \wedge t(p') < t(q') \;\Leftrightarrow\; \exists! p, q \in T : \\ (p' = p, q' = q) \wedge \; t(p) < t(q) \\ \textit{where } t(x) \textit{ is the temporal order of } x\end{gathered} \tag{3.4}$$

3.3.2 Observation

Our Method is derived from the following observation. In figure 3.5, there is a sample of GPS trajectory from our trajectory sensing project, and data points are divided into 20 groups with same points - `N`. We can draw minimum bounding rectangle (blue rectangle) called *MBR* on each group. In addition, we divide each MBR into cells of equal size (grey grid) to calculate the

Table 3.1: Information content and MBR area

MBR No.	**MBR Area**	**Sum of Information Content**
1	208.4	49.2
2	177.8	45.3
3	107.5	27.9
4	140.3	31.4
5	140.3	41.7
6	252.0	53.0
7	36.5	41.7
8	271.9	53.0
9	98.2	28.1

information content of MBR. Sum of information content of one MBR is calculated as:

$$\sum_{i=1}^{xn}\sum_{j=1}^{yn} -\log(p_{ij}) \tag{3.5}$$

Where p_{ij} is the frequency of data points within the cell[i,j] (see bottom-right part of figure 3.5).

Finally we calculated the area of each MBR and the sum of information content of each MBR as described in table 3.1. From table 3.1, we can find a strong positive correlation between MBR area and sum of information content. In fact, we calculate the correlation coefficient for them over huge real data and the value is up to about 0.8. Return to our goal of data reduction, it is to achieve the minimal data loss, which means keep information content as same as possible comparing to original data. Therefore, we claim this assumption: **The bigger the area size of MBR of IC (Minimum Bounding Rectangle of Information Content) is, the more the sampling points should be stored (see MBR 8 in figure 3.5). Otherwise, we can omit more sampling points (see MBR 7).**

Based on the analysis above, we developed a trajectory simplification method called *IC MBR* which consists of **divide and merge principle**

and **selection strategy** described in detail in section 3.3.3 and section 3.3.4 respectively.

3.3.3 Divide and Merge Principle

Based on the assumption stated in the previous section, the following principle is employed: we divide the bigger MBRs while merge the smaller MBRs so as to keep the nearly uniform size of MBR. There is an example shown in figure 3.6. Initially, 4 MBRs are drawn on it by every 4 points (the number is an input parameter given by user), and then merge MBR 2 and MBR 3 because their area is far less than the standard MBR, while split MBR 4 because it is far bigger than the standard MBR (*standard MBR* is an input parameter which is referred to comparing individual MBR). The size of the standard MBR can be obtained by users experience or specific requirements, which as well as its points number directly affect accuracy and compression ratio of trajectory that will be discussed in later section. Another technique to determine an appropriate standard MBR (called *adaptive MBR*) is to dynamically adjust the value by calculating area size within a tuning period (e.g. take the moving average value of all MBRs). To keep the consistent accuracy, an adaptive MBR is more effective in the case of multi-transportation mode, since the area is subject to variations depending on walk mode or driving mode. Assuming that the original sampling interval is a fixed time interval, standard MBR area is supposed to be assigned as a greater value in driving mode yet a less value in walking mode. Hence, if the area or points number of standard MBR is dynamically programmed with the consideration of both transportation mode and sampling interval, the result may be more satisfactory.

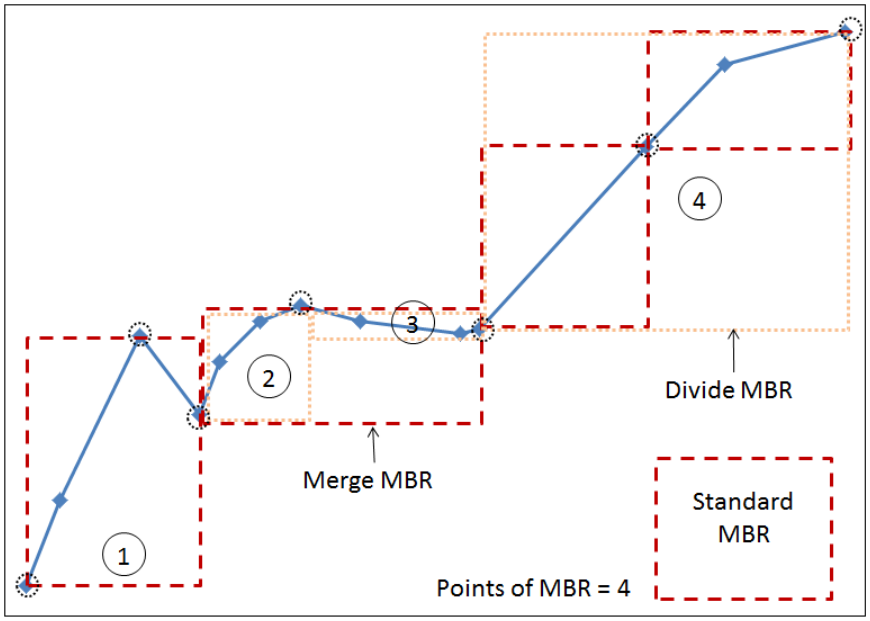


Figure 3.6: The illustration of *IC MBR* method

3.3.4 Selection Strategy

Through dividing or merging MBRs, every resulted MBR will contain the comparatively uniform information content. Hence, we take such a strategy to extract points based on such an assumption that points on the boundary of MBR contain more information content and it is advantageous to choose boundary points for the sake of lowering area error. As shown in table 3.2 (call *4-2-1-0.5 rule*), the points to be stored are determined by comparison with the standard MBR area. For instance, select 4 points on boundary of MBR when the MBR meets condition 2 (see table 3.2), select the first point and the last point when meets condition 3, and select the median point when meets condition 4. In the case of the condition 5, if the MBR is a divided MBR then select the median point; or merge the MBR (which is explained in section 3.3.3 and rule 1 of table 3.2).

After integrating the **divide/merge principle** with the **selection strategy**, algorithm of *IC MBR* method can be described as algorithm 1. This

Algorithm 1 *IC MBR* method

```
1: function DVIDE_MERGE(St_MBR_Pts_Num, St_MBR_Area, Traj)
2:     Num ← St_MBR_Pts_Num
3:     for all point in Traj do
4:         if Num = Buf.Count then
5:             rlt ← SelectPoints(Buf)
6:             if rlt is false then                          ▷ Merge MBR
7:                 Num ← Num * 2
8:             end if
9:         else
10:            Buf.Add(point)
11:        end if
12:    end for
13: end function
14: function SELECTPOINTS(Buf)
15:    Area ← CalcArea(Buf)                  ▷ MBR or Polygon area
16:    Learn(Area, St_MBR_Area)              ▷ Adjust standard MBR area
17:    if Area > St_MBR_Area * 2 then                    ▷ divide MBR
18:        SelectPoints(Buf/2)                          ▷ first half of Buf
19:        SelectPoints(Buf/2)                          ▷ second half of Buf
20:    else if Area < St_MBR_Area/4 then return false
21:    else                                             ▷ by selection strategy
22:        SavePoints()
23:    end if
24: end function
```

Table 3.2: Points selection strategy

No.	Condition	Selection Criteria
1	$MBR(N) \subset [St\_MBR * 2, \infty)$	Divide MBR
2	$MBR(N) \subset [St\_MBR, St\_MBR * 2)$	4 points: $x(min)$, $y(min)$, $x(max)$, $y(max)$
3	$MBR(N) \subset [St\_MBR * 0.5, St\_MBR)$	2 points: $x(0)$, $x(N-1)$
4	$MBR(N) \subset [St\_MBR * 0.25, St\_MBR * 0.5)$	1 point: $x(median)$
5	$MBR(N) \subset [0, St\_MBR * 0.25)$	0.5 point: Merge MBR or $x(median)$

Table 3.3: Time complexity comparison

Method Name	Normal Case	Worst Case
Uniform sampling	n/β	n/β
Dead reckoning	n	$2n$
Douglas-Peucker	$n\log(\beta)$	$n\beta$
IC MBR (Ours)	$n\log(\beta)$	$2n\log(n)$

method adapts bottom-up and top-down strategy simultaneously, which recursively approximate line segments within a rectangle. Incidentally, in the 15th line, area is calculated by minimum bounding rectangle or polygon, and in the later section we will discuss the performance of both of them. Besides, our method's time complexity is $O(n/\beta \cdot \beta\log(\beta)) = O(n\log(\beta))$ where β is the point number of buffer. In table 3.3, we compare our method with conventional methods (adjusted to online process) in terms of time complexity, and in normal case our method is the same as *Douglas-Peucker* method which is the favorable method in many fields.

3.4 Evaluation Methods

To evaluate the accuracy of each method in terms of perpendicular distance (PD), synchronized Euclidean distance (SED) and enclosed area (EA),

we implement *IC MBR* method, *uniform sampling* method, *dead reckoning* method and *Douglas-Peucker* method which is slightly modified to adapt to online processing. It is important to note that even though DP method is an offline method, we add a buffer for local processing so that we can compare these methods under fair conditions. Obviously, this additional parameter may affect the initial performance to some extent while improve the time cost.

3.4.1 Two Conventional Error Metrics

Two conventional error metrics: average perpendicular distance and average synchronized distance are uniformly defined as:

$$m_{1,2}(T^o, T^s) = \frac{1}{N}\sum_{i=1}^{N} d_i^2 \tag{3.6}$$

Here, N is the total points of original trajectory. *PD* refers to perpendicular distance between the point of original trajectory and the line segment of simplified trajectory (see solid arrows in figure 3.4), which is obtained by the following equation:

$$d_i = \frac{|(x_{s(k+1)} - x_{s(k)})(y_{s(k)} - y_i) - (y_{s(k+1)} - y_{s(k)})(x_{s(k)} - x_i)|}{\sqrt{(x_{s(k+1)} - x_{s(k)})^2 + (y_{s(k+1)} - y_{s(k)})^2}} \tag{3.7}$$

where $P_{s(k+1)} = (x_{s(k+1)}, y_{s(k+1)}) \in T^s, P_{s(k)} = (x_{s(k)}, y_{s(k)}) \in T^s$ and $P_i = (x_i, y_i) \in T^o$

SED calculates the distance from original point to virtual simplified point which is at identical timestamp (see dot circles in figure 3.4). It considers the temporal attribute of the point sequence, thus it is thought as a better error metric. The virtual simplified point is missing in simplified trajectory,

Figure 3.7: A real GPS trajectory with large area displacement

whereas it can be obtained as follows:

$$P_i = P_{s(k)} + \frac{P_{s(k+1)} - P_{s(k)}}{t_{s(k+1)} - t_{s(k)}} t_i \tag{3.8}$$

3.4.2 New Error Metric

Although *PD* and *SED* are widely used in the literature, there are two drawbacks as follows:

1. As stated in section 3.2, in some cases, the distance-based error is small while the area error is huge. Hence, it will mismatch the real trajectory and simplified trajectory.

2. In addition, the point of trajectory is not an accurate position since GPS accuracy is uncertain. If one point has an unexceptional change due to GPS uncertainty, distance-based error is subject to this sort of burst error while area metric is robust from the viewpoint of uncertainty tolerance.

Therefore, we introduce a continuous 2-dimensional error metric - *enclosed area*.

GPS Errors Analysis

We will discuss the GPS error sources to justify the necessity of our new error metric. The accuracy of GPS depends on a complicated interaction of various factors.

To analyze the effect of errors on accuracy, a fundamental assumption is usually made that the error sources can be allocated to individual satellite pseudo-ranges. The effective accuracy of the pseudo-range value is termed the user-equivalent range error (UERE). There are the major error sources in GPS which develop error budgets for UERE: satellite clock error, ephemeris error (position of satellites), atmospheric effects (ionospheric and tropospheric delay), receiver noise and resolution, and multipath/shadowing effects [KH06].

The position error that results from UERE depends on the user/satellites relative geometry, which is called geometric dilution of precision (GDOP). If the satellites viewed from user location is close together (or in a line), the GDOP factor will be higher. Loosely speaking, error in the GPS solution is estimated by the formula[KH06]:

$$\sigma_p = GDOP \cdot \sigma_{UERE} \tag{3.9}$$

where σ_p is the standard deviation of the positioning accuracy, σ_{UERE} is the standard deviation of the satellite pseudo-range measurement error and GDOP is the geometry factor of the satellites for a specific location and time of day.

Usually, the error components are considered independent, and the composite UERE for a satellite is approximated as a zero mean Gaussian random variable where its variance is determined as the sum of the variance of each of its components. UERE is usually assumed to be independent and identically distributed from satellite to satellite. However, the position error in a consecutive trajectory is not fully independent. As we see from the error factors described above, GDOP and multipath depend on the specific location, which are the major errors in a short time period while other factors don't fluctuate wildly in a small space-time and be compensated by calibration to some extent. If a user moves in a street, then the GDOP factor and multipath

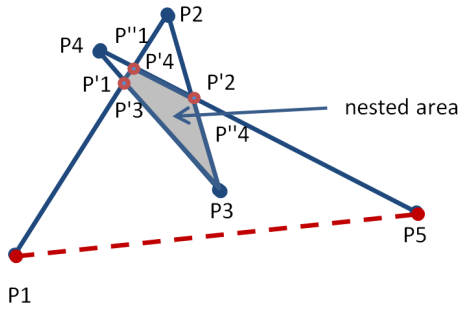


Figure 3.8: A self-intersecting polygon

effect caused by blocking by surrounding buildings will be almost constant unless the surrounding is changed abruptly. That is why we assume the GPS error follows Gaussian distribution in later simulation, whereas we claim the case of figure 3.4 is necessary to be solved yet not be properly answered by existing error metrics. In fact, we found there are considerable cases (i.e. trajectory with large area displacement yet small distance displacement) from our collected trajectories. Figure 3.7 shows a GPS trajectory in which the blue trajectory is obtained by GPS while the light-yellow one is the real moving trajectory. As the buildings along the road is homogeneous, the GDOP and multipath effect are identical, which resulted in the same distance error. Of course, this sort of case is common in urban area while the situation is different in other area.

Enclosed Area

Enclosed area which is a polygon confined by original trajectory and simplified trajectory (the dashed area of figure 3.4). Although EA is obviously advantageous, its calculation is quite troublesome [NP82][PS06] provided that the polygon contains self-intersection (see figure 3.8). Nevertheless, we devised such an algorithm to solve this problem (see algorithm 2):

(1) The intersection point is obtained by geometry formula.

(2) If exist cross point, it will be sequentially inserted into a list which con-

sists of original points, but do not insert two or more intersection points on the same line segment. Take the figure 3.8 as an example, finally the list in algorithm 2 will be $(P1, P'1, P2, P'2, P3, P'3, P4, P'4, P5, P1)$.
(3) The sub sequence of the list split by intersection point is a line segment or a simple polygon whose area can be easily obtained as follows - by *outer product of vector*:

$$Q_j = \frac{1}{2} \mid \sum_{i=1}^{K} \overrightarrow{p_i} \times \overrightarrow{p_{i-1}} \mid \tag{3.10}$$

According to this algorithm, the nested area (see figure 3.8) will be accumulated two or more times, whereas it is reasonable. Then, we formally define the third error metric - average enclosed area as:

$$m_3(T^o, T^s) = \frac{1}{N} \sum_{i=1}^{J} area(Q_j) \tag{3.11}$$

Uncertainty Tolerance

GPS system has intrinsic error of approximately from several meters to tens of meters. For a specific location, the inaccuracy is not constant and it can be seen as normal distribution, which means some particular points reach large displacement while most points are within slight error range (but in commercial GPS system these slight errors may be eliminated by smoothing techniques). Note that the errors in a series of consecutive locations within a short time are dependent to a large extent as analysed in section 3.4.2. We want to explore the different effect between distance-based metric and area-based metric while taking the GPS uncertainty into account.

As shown in figure 3.9, $P_1 \cdots P_i \cdots P_N$ is the GPS points of original trajectory, and the dot line between P_1 and P_N is the simplified trajectory. In addition, we assume P_k has a large error whose corresponding real position is P'_k, while other points have only slight error. We define uncertainty tolerance which indicates how significant the uncertain (wrong) position affects

Algorithm 2 Area Calculation of Arbitrary Polygon

```
1: function CalcAreaOfPolygon(Polygon, PointsNum)
2:     for all point p_i in Polygon do
3:         List.Add(p_i, 0)                          ▷ 0: original point
4:         for j = 0; j < i − 1&&i > 1; j++ do
5:             P ← CrossPoint(p_i, p_{i+1}, p_j, p_{j+1})
6:             if !List.Find(p'_i) then                          ▷ after p_i
7:                 List.Insert(p'_i ← P, 1)              ▷ 1: cross point
8:             end if
9:             if !List.Find(p'_j) then                          ▷ after p_j
10:                List.Insert(p'_j ← P, 1)              ▷ 1: cross point
11:            end if
12:        end for
13:    end for
14:    anchor ← 0
15:    for i := 0 → List.Count do
16:        if List[i].flag == 1 then
17:            CalcAreaOfSimplePolygon(List.Range(anchor, i))
18:            anchor ← i
19:        end if
20:    end for
21:    CalcAreaOfSimplePolygon(List.Rage(anchor, i))
22: end function
```

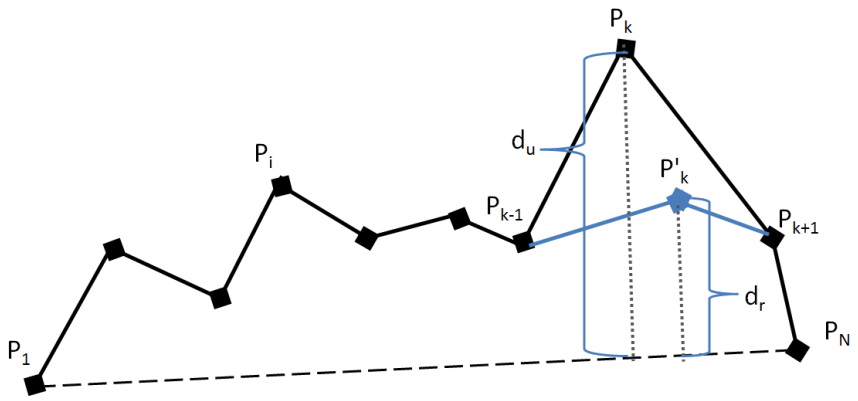


Figure 3.9: Error metric under GPS uncertainty

the simplification process. For distance-based simplification, the uncertainty tolerance is $T_d = \frac{|d_u - d_r|}{d_r}$, where d_u is the distance from GPS point to simplified trajectory, and d_r is the distance from real position to simplified trajectory. On the other hand, the uncertainty tolerance of area-based simplification is $T_Q = \frac{|Q_u - Q_r|}{Q_r}$, where Q_u is the area enclosed by GPS points, and Q_r is the area enclosed by points of real position. In figure 3.9, Q_u is the area of polygon $(P_1, \cdots, P_i, \cdots, P_{k-1}, P_k, P_{k+1}, P_N)$, and Q_r is the area of polygon $(P_1, \cdots, P_i, \cdots, P_{k-1}, P'_k, P_{k+1}, P_N)$. For simplifying calculation, here we assume GPS points with slight errors as real position while distinguish the point (i.e. P_k) with the largest error from real position (i.e. P'_k). Note that, the higher the value of uncertainty tolerance, the weaker it is against uncertainty. In figure 3.9, suppose there are only 3 points (P_1, P_k, P_N), then $Q_u = \frac{1}{2} d_u |P_1 P_N|$ and $Q_r = \frac{1}{2} d_r |P_1 P_N|$. The uncertainty tolerance T_Q is $\frac{|Q_u - Q_r|}{Q_r} = \frac{|d_u - d_r|}{d_r} = T_d$, which indicates distance-based metric performance as same as area-based metric in the case of 3 points.

We conduct the experiment by simulating 10,000 times for each pattern (combination of GPS points and GPS error). Figure 3.10 shows the result. The vertical axis (Z) is the value of $(T_d - T_Q)$. We can find in the case of 3 points the difference is zero which is consistent with our theory. In generally, distance-based metric performs worse in terms of uncertainty tolerance,

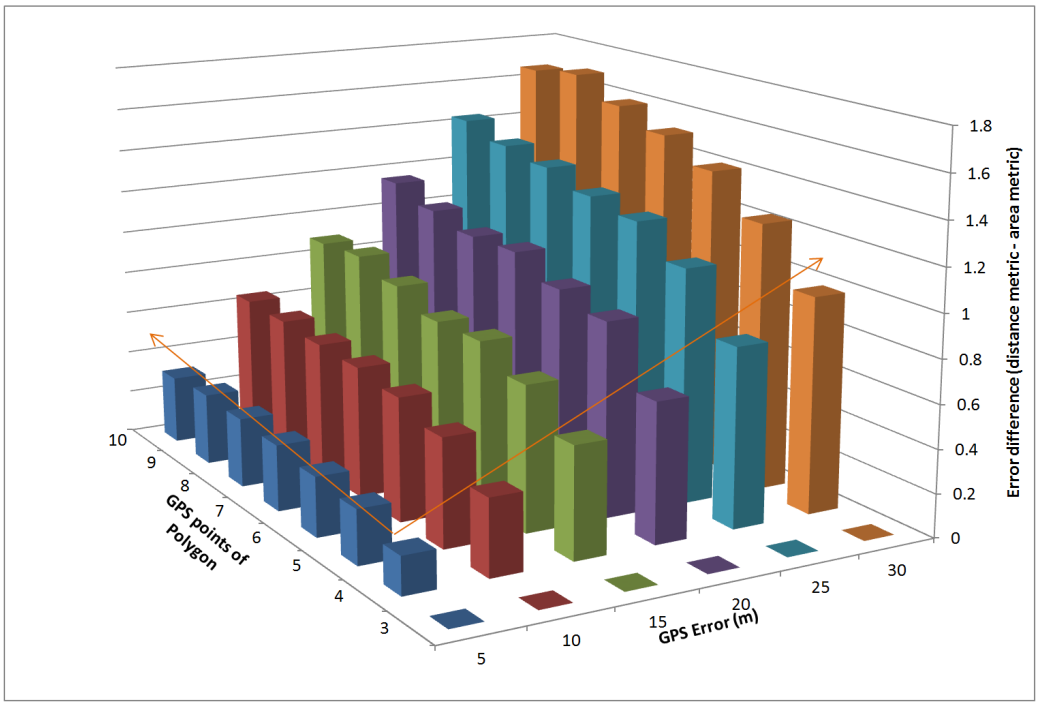


Figure 3.10: Difference of uncertainty tolerance

and as the GPS error increases, the performances of distance-based metric deteriorates. The same trend can be seen in terms of GPS points of polygon.

3.5 Experimental Results and Discussion

In this work a series of experiments are conducted by using the data collected through our *trajectory sensing* project. Aside from it, we also make use of Microsoft GeoLife [ZLCea08][ZZXea09] dataset that consists of 178 users in a period of over four years (from April 2007 to October 2011). Various transportation modes are included in the data set, including walking, driving, train travel and etc. Experimental data files are selected by different file size, different transportation modes and different trajectory shapes so that we can compare the performance to draw a general conclusion.

3.5.1 Performance Comparison

Trajectories are compressed with 3%, 10%, 20% and 50% and are measured by 3 error metrics (namely *EA*, *PD* and *SED*). Here, compression ratio (*CR*) is defined as the number of original points divided by the number of compressed points. From figure 3.11 (the results of compression ratio 10% and 20% are shown while other compression ratio results are skipped, but a complete comparison is described later), we can find that (1) accuracy gets worse as compression ratio decreases; (2) accuracy also greatly varies due to different trajectory files which mean different shape of trajectories; (3) uniform sampling produces an uncertain output, in other words, its performance drastically fluctuates. In addition, our method can meet different compression ratios and error tolerances by adjust the parameters (points of standard MBR and its area). Generally speaking, these two parameters are similar to distance threshold in *DP* or *DR* method, that is, lessening the value of them will earn better accuracy but high compression ratio.

Figure 3.12 shows the normalized error (the value is scaled to [0,1]) of *EA*, *PD* and *SED* in different compression ratio. As a result, our method holds an absolute advantage in terms of *EA* metric and competitive performance in *SED* metric but poor performance in *PD*. The reason is that we measure displacement directly by enclosed area, and there is an inherent relation between area and distance. Our method filters point based on information contents that is measured by area in 2-dimensional plane, which resulted in a good performance in *EA*.

3.5.2 Parameters Analysis

In our method there are two important user-specified parameters which are the points number of standard MBR and the area of standard MBR. Besides the area of standard MBR is also subject to the adaptive MBR or fixed MBR. Since the performance is significantly affected by these factors, we

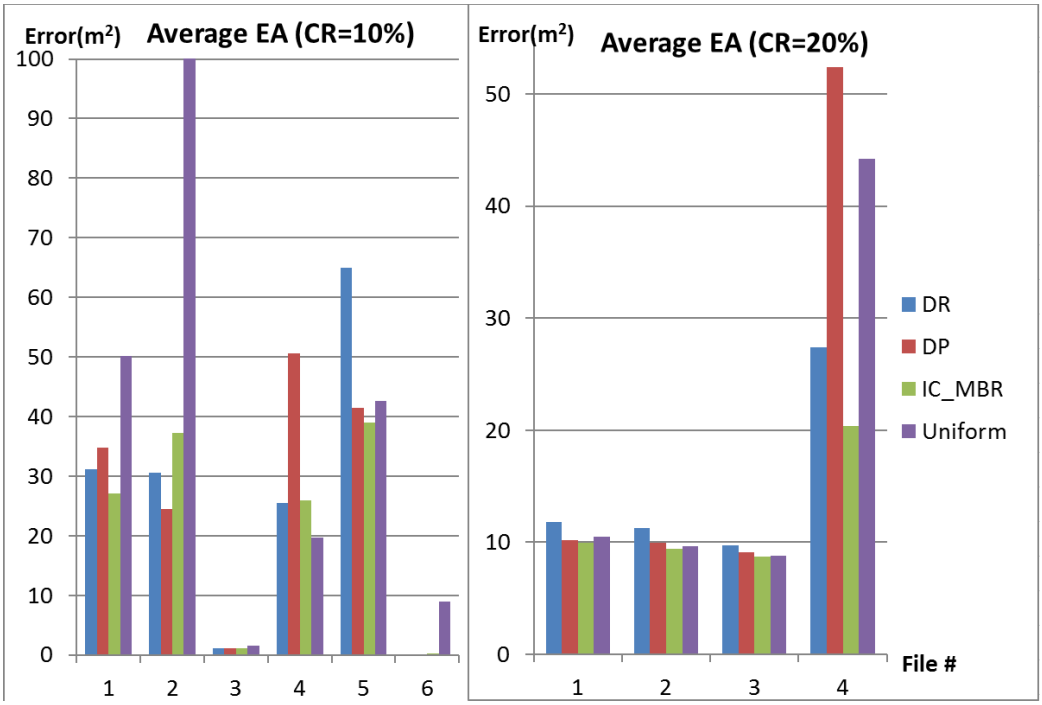


Figure 3.11: Average *EA* with 10% and 20% compression ratio

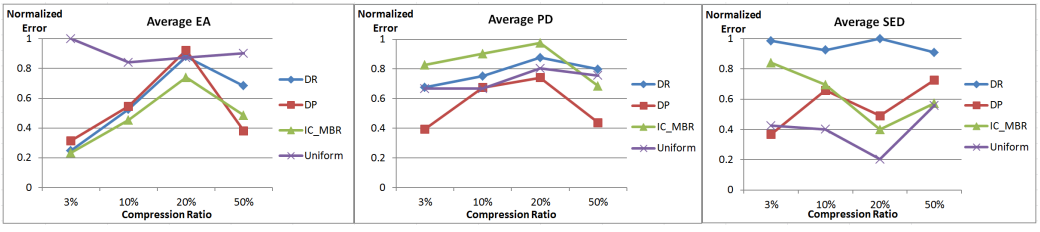


Figure 3.12: Normalized error of *EA*, *PD*, *SED*

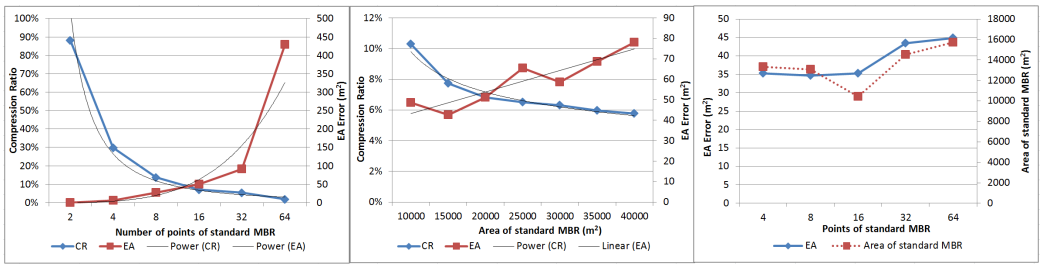


Figure 3.13: Parameters effect over accuracy

will explore the relationship between them and *EA* accuracy.

In the left part of figure 3.13, it employs the moving average to adjust standard MBR area. We can find: as the points of standard MBR increases, the *EA* error increases and the compression ratio decreases simultaneously. In fact, there is a positive linear relationship between the points of standard MBR and *EA* error but a negative linear relationship between the points of standard MBR and compression ratio. (Note that although the figure shows a power relationship, actually it is linear relationship since the horizontal axis is also power-scaled.) Hence, we need a trade-off between compression ratio and *EA* error, and in most cases the number of points of standard MBR ranging from 8 to 20 is advisable.

In the middle part of figure 3.13, we fix the points of standard MBR as 16 but adjust standard MBR area manually. It shows that the compression ratio slightly decreases as the area of standard MBR increases. Besides, the *EA* fluctuates along with area of standard MBR. However, in a real-time mode it is hard to determine the optimal area of standard MBR to get the smallest error.

In the right part of figure 3.13, we observe the change of *EA* with parameters in adaptive adjustment mode when the user specifies the compression ratio. From the figure (here *CR* is fixed at 10%), we find that along with the increase of points of standard MBR, the *EA* error increases. At the same time, the area of standard MBR increases too.

Our method is based on area metric, whereas we calculate not the area

of polygon (the actual enclosed area) but the area of MBR instead during divide/merge MBR. The reason is that there is a strong correlation between polygon area and its corresponding MBR area. In addition, to calculate polygon area is much more computation-intensive than to calculate MBR area. In fact, the time complexity of our method with polygon calculation is: $O(n/\beta * \sum_{j=1}^{\beta} \frac{j(j+1)}{2}) = O(n/\beta * \frac{1}{2}(\frac{\beta(\beta+1)}{2} + \frac{\beta(\beta+1)(2\beta+1)}{6})) = O(n * \beta^2)$ where β is the point number of buffer. Comparing to the method with MBR calculation ($O(nlog\beta)$), the scale of time complexity is different. We actually compared these two ways, and find that by calculating polygon area the accuracy can be improved by `2.7 times` while the computation time may increase by `34.3 times`. Due to the requirement of real-time simplification, we choose the simple way - MBR calculation - but sacrifice the accuracy.

In addition, we calculated the correlation of MBR area and information content of MBR, to explore its relation with the *EA* accuracy. As a result, there is no strong correlation between them, namely the strong correlation of MBR area and information content of MBR does not necessarily indicate good accuracy. Note that this result does not go against our method's efficiency since our method is finally evaluated by error metrics as above.

3.6 Chapter Conclusion

In this chapter, we proposed a novel scheme: `divide/merge principle` and `selection strategy` to reduce data for spatial trajectory. To measure displacement correctly, we newly introduced enclosed area metric which is proven more robust against GPS uncertainty. Although *DP* method still outperforms other methods from the perspective of whole performance (by comparing the average value of all 3 metrics), our method is the most efficient method in terms of *EA* metric - the most convincing measure. Furthermore, our method is a pure online procedure which can be readily installed at the mobile terminal to preprocess trajectory before sending it to back-end server.

On the other hand, the transformed online DP method needs a big enough buffer to guarantee the accuracy and compression ratio.

However, our method as well as existing methods merely takes geometric feature (linear or areal displacement) into account, which may lead to the loss of other information (e.g. speed). Besides, in the second component of our method (i.e. selection strategy), we just intuitively save boundary points for achieving the least areal displacement. Consequently, this selection can't exhibit the optimal simplification and it is sensitive to the shape of trajectory.

In the future, we consider extending the MBR of IC idea to Minimum Bounding n-dimensional Cube so as to compress multidimensional trajectory. Furthermore, it would be extremely challenging and meaningful to explore the relationship between the accuracy and the features of trajectory, eventually to seek optimal input parameters (points of standard MBR and area of standard MBR) and better selection strategy.

Chapter 4

Sensor Data Reduction: REPSense

4.1 Introduction

Today's smartphones are programmable and come with a wide variety of inexpensive but powerful embedded sensors, such as accelerometer, GPS, gyroscope, microphone, light sensor, camera and Wi-Fi detector. The ubiquitousness of mobile terminals and the development of sensors have opened up a new avenue to comprehensively sense, learn and share information about our lives and the world around us. In the domain of mobile phone sensing, there are countless fruitful studies on human behaviour detection, environment monitoring, traffic monitoring, social interaction and so on [KXAA13]. In our project, called *trajectory sensing*, participants collect data (including ambient noise, light, location, etc.) on their commuting routes by using their smartphones. In this approach of sensing, commonly known as participatory sensing [BEHea06] or crowd sensing [GYL11], data is collected in a distributed and loosely controlled way. It often results in data redundancy, which leaves room for identification and removal of the redundant sensor data.

Another motivation for data reduction is to improve performance. The huge volume of data may seriously hamper the performance of a variety of applications. Massive data creates multiple problems related to storage, transmitting cost and power consumption etc. The foremost issue is that the enormous volume of data can easily overwhelm further computing and human analysis. For example, clustering performance (e.g., accuracy, speed etc.) would exponentially decrease in proportion to the number of sampling data [CLLea11]. We faced this problem in our *trajectory sensing* project. In that project, we collected ambient noise and light from different areas at night via mobile sensing to establish a relation between noise, light and night-life activity of a region. We found that data processing time is an obstacle when computing data similarity among different regions' ambient noise-light. Furthermore, some powerful classification methods lose their abilities in face of huge data sets[CLYea08].

The above restrictions call for the need to reduce data volume. A straight forward way would be to use conventional compression methods based on bit-rate reduction, such as *LZ* (used in zip) or *DCT* (Discrete Cosine Transformation). However, these compression methods cannot process data-stream in real time, and are computationally intensive when implemented on WSNs or smart-phones. Hence, for mobile sensing or sensor networks, an array of dedicated methods[SM11], [JKGS12], [NYZea12] have been proposed to handle this issue. These methods mainly focus on solving storage, transmission and power consumption related issues. However, they fail to address two other issues: 1) maintaining the data diversity and 2) compressing the data in such a way that would allow applications to perform computation on the reduced data without uncompressing it. We address both these issues in the proposed method.

Our objective in this work is to reduce data volume, and at the same time, make sure that compressed data can be directly used by subsequent applications without uncompressing. In addition, we expect the reduced data

to represent the whole sample as much as possible. From the view point of information theory, we hope that after data reduction, output data retains information content as close as possible to the original data. In other words, by using the reduced data application should be able to reach the same conclusions they would have reached had they used the original data. The essence of this problem has analogy to electoral systems in which congressmen are elected to delegate public opinion. There are two popular voting systems: *proportional representation* (*PR*) and *single-member district* (*SMD*). The former is considered as the fundamental of representative democracy and is used in many countries, while the latter is widely used in USA, UK, etc. Considering the pros and cons of each of them, some countries (e.g. Japan) have combined these two systems; with the consideration of opinion diversity, minorities should be assigned more weight to be elected. It is intuitive that a combination of *PR* and *SMD* is more beneficial because it can balance the majority and the minority by compromising majority decision with minority interest. Motivated by this, we have developed our method based on this principle.

Our contributions can be summarized as follows:

- We defined a new data model called *REPresentative Sense* (*REPSense*) to represent data from the viewpoint of *data diversity*. We then developed a divide/merge scheme to implement this model which can compress multiple-dimensional data with arbitrary distribution in real-time.

- Based on large real data set collected in our project, a series of experiments were conducted to evaluate the performance of our method in terms of data divergence and data classification performance. We also applied it on a data mining application, and we found that the proposed method is more effective compared to a set of state-of-art baseline methods.

4.2 Related Work

4.2.1 Mobile Sensing

Mobile crowd sensing [GYL11] or participatory sensing [LEM+08] which applies crowd-sourcing approach has many merits, for instance, (1) with the participation of public users, we can collect data within a wider area and with better quality; (2) it also brings down cost since all participants are basically volunteers. There are multiple fruitful applications, such as data collection framework [RES10] and environment monitoring (NoiseTube[10N]). It is undoubtedly important to seamlessly combine trajectory tracking with crowd sensing so that we launched a project called *trajectory sensing*. In most cases, the sensor data needs to be associated with the spatio-temporal attribute for further exploration. For example, it would be meaningless if the temperature or ambient light is gathered without the label of time and location. Moreover, people's daily-life trajectories can be used to understand human activities [ZLCea08].

4.2.2 Data Reduction in Sensing

Although sensor data is useful, vast data inflicts heavy burden on applications. Hence it has enjoyed much research attention. This issue becomes severer and unendurable in mobile devices and sensor networks. Nikzad et al.[NYZea12] presented a technique to reduce the amount of data in smartphone for producing pollution map. Chu et al. [CDHH06] proposed a technique to aggregate data by probabilistic approximation, which is claimed to be well suited to anomaly- and event-detection applications. As Lane studied in [LXLea11], there exists similarities among sensor data, which indicates the redundancy in sensor data can be eliminated without jeopardizing applications.

Aside from these dedicated methods for particular applications, there are

a number of general methods. We can categorize them into 3 groups in terms of compression granularity.

(1) **bit-rate compression**

Compressive sensing[BR07] is a universal method to compress data. However, compressive sensing is computationally expensive while calculating a transformation matrix, and it is available only when the target data can be transformed into a sparse matrix. In fact, toward our trajectory sensing data, it is inefficient to compress at large ratio. Moreover, compressed data need to be uncompressed for using, which deviates from our goal that we expect to directly apply data after reducing the amount of data. For the same reason, plus the need of on-line processing, conventional *bit-rate* data compression methods, e.g. *Zip* method, *DCT*, and *DWT* (Discrete Wavelet Transform) cannot be considered either.

(2) **field-rate compression**

Stratified sampling[Ney34] which is a *field-rate* method, divides total population into subgroups (called *strata*) and then select sampling points by proportionate allocation; nevertheless, it can be explained as a special case of our method.

(3) **dimension-rate compression**

Another solution to data reduction is to reduce data dimensionality; the typical method is *principle component analysis*[Jol86] which uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables.

In our work we focus on field-rate compression since its result can be directly used for further analysis which meets our goal. Most widely used methods are *uniform sampling* and *dead reckoning*, which are our baseline methods. *Uniform sampling* is a simple method which sparsely selects the points to store in every given time interval but discards other points. Although uniform sampling may provide a simple and cost-effective solution, it is insensitive to sensor readings due to its ad hoc nature. *Dead reckoning*

(*DR*) is a localized processing routine which make use of the characteristics of the immediate neighbouring points in determining whether to retain the current point. Namely, it will only store points which are varied from the anchor point by over a predefined threshold. Hence, it will heavily destroy the original data distribution. Aside from these methods, we also need to consider the state-of-art methods. Raza et al.[RCMea12] proposed a method called *Derivative-Based Prediction* (*DBP*) to reduce the quantity of data reports in wireless sensor networks. *DBP* takes two steps, in first step it learns data sampling to generate a linear model which is obtained by derivative-form and store its last point during learning. In second step it uses derived linear model to predict data, and go to first step to rebuild prediction model if newly generated data is out of prediction model in terms of both value tolerance and time tolerance.

4.3 Proposed Scheme

4.3.1 Problem Statement

The ultimate goal of our project is to learn human activities under particular physical environment by sensing and then to discover knowledge. However, the problem we plan to solve in this work is data reduction in real-time before transmitting or data reduction off-line before data analysis (see figure 2.4). There is a wide variety of sensors built in smartphone. In our *trajectory sensing* project, we select *GPS*, ambient light sensor, microphone, air pressure sensor and accelerometer.

During our sensing experiments (described in chapter 2) and later data analysis, we find the data volume causes an array of problems. Because participants are sensing environment while they are fast moving (e.g., by bike or train), sampling period is supposed to be very short, so sensor is capable of capturing signal changes, meaning the data volume will accumulate more

quickly than in other applications. Therefore, the aforementioned obstacles of vast data, including storing, transmitting, power consumption and data analysis efficiency, become more notable and unendurable in our project.

After data reduction, the amount of data points decrease but data values and the order are never changed, that is, reduced data meets equation (4.1) and (4.2) (It is introduced in chapter 3 while state here again for conveniences). We call these two properties *value-invariance* and *order-invariance* respectively.

$$\forall p' \in T' \;\Rightarrow\; \exists! p \in T : p' = p \tag{4.1}$$

$$\begin{gathered} \forall p', q' \in T' \wedge t(p') < t(q') \;\Leftrightarrow\; \exists! p, q \in T : \\ (p' = p, q' = q) \wedge \; t(p) < t(q) \\ \textit{where} \;\; t(x) \textit{ is the temporal order of } x, \\ T \textit{ is the original data, } T' \textit{ is the reduced data} \end{gathered} \tag{4.2}$$

4.3.2 Observation

To reduce sensor data, we rethink the problem of data reduction from the following viewpoint. Although it is generally assumed that data loss is ineluctable after reducing data, reduced data is still expected to represent the original data as parliament members are elected to represent all citizens. Under majority rule of democratic nations, representatives are assigned in proportion to the number of supporters, which is the basis of proportional representation (*PR*). On the other hand, for the sake of protecting minority right, a priority or weight should be given to select minority representative for meeting opinion diversity, which reaches the same effect as single-member district (*SMD*). In our method, we balanced these two principles.

In election, nation is divided into many constituencies (usually by state or prefecture) which consist of different interest groups, e.g., middle class, rich and poor. In analogy to it, as shown in figure 4.2 which is boxplots of the same data shown in figure 4.1, the total population is divided into

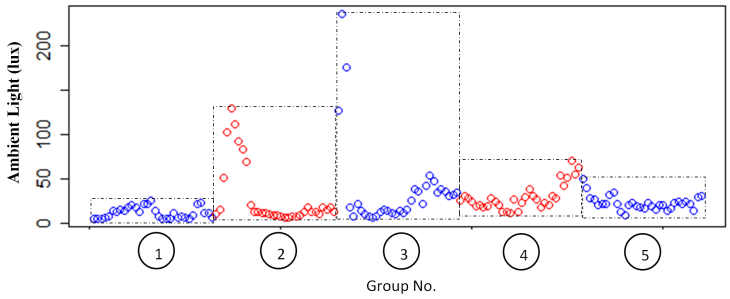


Figure 4.1: Sample data of ambient light

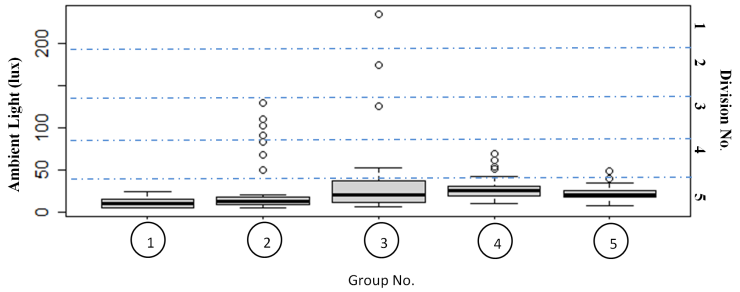


Figure 4.2: Box-plots with division of sample data

5 groups (horizontal axis) as constituencies, and it is also divided into 5 divisions (right vertical axis) as interest groups. Note that, dividing into many groups is mainly because our method aims to process data on-line so that we handle data group by group (a group means a local buffer). Next step is to select points to represent a group, which is similar to vote representative in election. Based on *proportional representation*, in figure 4.2 many points from 5th division will be selected, on the other hand, minority division (such as 1st, 2nd division) can also be picked out if adopting *single-member district*. With consideration for opinion diversity, a combination of these two systems seems favorable.

In our case, we are concerned with *data diversity*. In figure 4.1, data points are divided into 5 groups each having N points. We can draw a *MBR (Minimum Bounding Rectangle)* on each group (dot dash rectangle in figure 4.1), and calculate the area of each MBR and the sum of information

Table 4.1: Information content and group area

Group No.	1	2	3	4	5
Group Area	635	3945	7322	1902	998
Sum of Information Content	3.58	8.20	7.00	4.27	1.60

content of each MBR as described in table 4.1. *Data diversity*, i.e., the sum of information content of each MBR (group) is calculated as:

$$\begin{gathered}\sum_{i=1}^{m} -\log(p_i),\ p_i\ is\ the\ frequency\ of\ data\ points \\ within\ i^{th}\ division\ where\ the\ entire\ group\ is \\ evenly\ divided\ by\ m = \sqrt{N}\end{gathered} \tag{4.3}$$

Data diversity is similar to species diversity, such as *Shannon index* or *Simpson index*; however, we remove the proportion factor from *Shannon index* to define our *data diversity*, because outliers are extremely valuable in most applications but they have a small proportion.

4.3.3 REPresentative Sense

Based upon above observation, we establish this principle: **data points are selected out according to both proportion and preference**. Here, preference refers to high priority on minority points which is analogous to minority nationalities. Minority points in our case are those points which deviate from centre of clusters. Since our basic idea comes from representative principle of democratic election, we call this method *REPresentative Sense*(*REPSense*). A factor for minority in i^{th} division is identified as this equation:

$$\begin{gathered}I(i) = \frac{-\log(p_i)}{\sum_{i=1}^{m} -log(p_i)} \\ where\ m\ is\ the\ number\ of\ divisions\ in\ group\end{gathered} \tag{4.4}$$

Accordingly, proportion of selected points from i^{th} division (namely *probability density function*) is:

$$Q_i = w * I(i) + (1 - w) * p_i \tag{4.5}$$

Where w is the weight assigned to preference of minority points, in a group G_j, p_i is obtained by:

$$\begin{gathered} p_i = \frac{|\{q|q \in [i-1,i) * div\}|}{N}, \\ where\ N = |G_j|, G_j = \{q_1, q_2.., q_n\} \end{gathered} \tag{4.6}$$

Here, division range (div) can be roughly determined by $(max(G_j) - min(G_j))/N$. However, in real application, we implement an adaptive fashion to adjust range with respect to variance of individual group and uniformity of total population. During a tuning period, a range is obtained by a $k-order$ moving average based on *Freedman-Diaconis rule* which is used to select bin size in a histogram:

$$\begin{gathered} div = \frac{1}{k} \sum_{j=1}^{k} \frac{Q_3(G_j) - Q_1(G_j)}{\sqrt[3]{N}} * 2, \\ where\ Q_1, Q_3\ are\ the\ 25^{th}\ and\ 75^{th}\ percentile \end{gathered} \tag{4.7}$$

Then, the number of division - m in group is:

$$m = 1 + \lceil \frac{max(G_j) - min(G_j)}{div} \rceil \tag{4.8}$$

After data reduction by any method, the distribution of original data (which is subject to $P = \{p_i\}$) is changed into another distribution, here which is subject to $Q = \{Q_i\}$. According to *Gibbs' inequality*:

$$\begin{gathered} H(P) = -\sum_{i=1}^{m} p_i \log(p_i) \leq -\sum_{i=1}^{m} p_i \log(Q_i) \\ = L(Q),\ where\ \sum_{i=1}^{m} p_i = 1, \sum_{i=1}^{m} Q_i = 1 \end{gathered} \tag{4.9}$$

$L(Q) - H(P)$ is defined as *Kullback-Leibler divergence* which is always more than 0. From here, we learn that any data reduction definitely cause information loss, therefore, we strive for lowering this sort of loss, which is evaluated in later.

4.3.4 Algorithm Description

Our algorithm for sensor data reduction employs such a scheme: first step is **dividing/merging groups**, and second step is **selecting points in resulted group** (see the detail in algorithm 3). Originally group size is identical while different groups possess different information content sum. As shown in table 4.1, group 2 and group 3 possess very high information content while information content of group 5 is too low. Consequently, a group will be resized to keep almost uniform information content, which is similar to creating voting district by the number of voters. There are two advantages for it: (1) it can raise efficiency through reducing groups which means reducing times of second step, i.e., selecting points; (2) it can guarantee the time tolerance since group size is referred to sensing duration.

For each point in a group derived from the first step, in the second step we can obtain the Q_i. Since Q_i is the probability (weight) to save, a *Bernoulli trial* with probability Q_i/p_i (i.e., $B(1, Q_i/p_i)$) is executed to determine whether to save it or not. Note that, in the 15th line of algorithm 3, the magnitude of group can be calculated by **area of MBR or sum of information content**. In the later section of this chapter we will discuss the performance of both of them. In addition, in the 16th line, we used a simple method to learn the data change in order to adjust the division (*div*) until it converges.

In equation (4.5), if w is assigned to 0, the method may regress to *stratified sampling* (similar to *PR* in electoral system), on the other hand, if w is assigned to 1, the method may be close to *dead reckoning* (similar to *SMD*).

Algorithm 3 *REPSense* method

```
1: function DIVIDE_MERGE(St_Group_Pts_Num, St_Group_Area, Traj)
2:     Num ← St_Group_Pts_Num
3:     for all point in Traj do
4:         if Num = Buf.Count then
5:             rlt ← SelectPoints(Buf)
6:             if rlt is false then                          ▷ Merge Group
7:                 Num ← Num * 2
8:             end if
9:         else
10:            Buf.Add(point)
11:        end if
12:    end for
13: end function
14: function SELECTPOINTS(Buf)
15:    Area ← CalcArea(Buf)                                  ▷ Group Area or IC Sum
16:    Learn(Area, St_Group_Area)                            ▷ Adjust division range
17:    if Area > St_Group_Area * 2 then                      ▷ divide Group
18:        n ← Buf.Count
19:        SelectPoints(Buf[0, n/2))                         ▷ 1st half of Buf
20:        SelectPoints(Buf[n/2, n))                         ▷ 2nd half of Buf
21:    else if Area < St_Group_Area/4 then
22:        return false
23:    else
24:        for all point in Buf do
25:            Calculate Q_i, P_i                            ▷ by equation (??),(??),(??)
26:            r = B(1, Q_i/p_i)                             ▷ by Binomial sampling
27:            if r = 1 then
28:                SavePoint(point)
29:            end if
30:        end for
31:        return true
32:    end if
33: end function
```

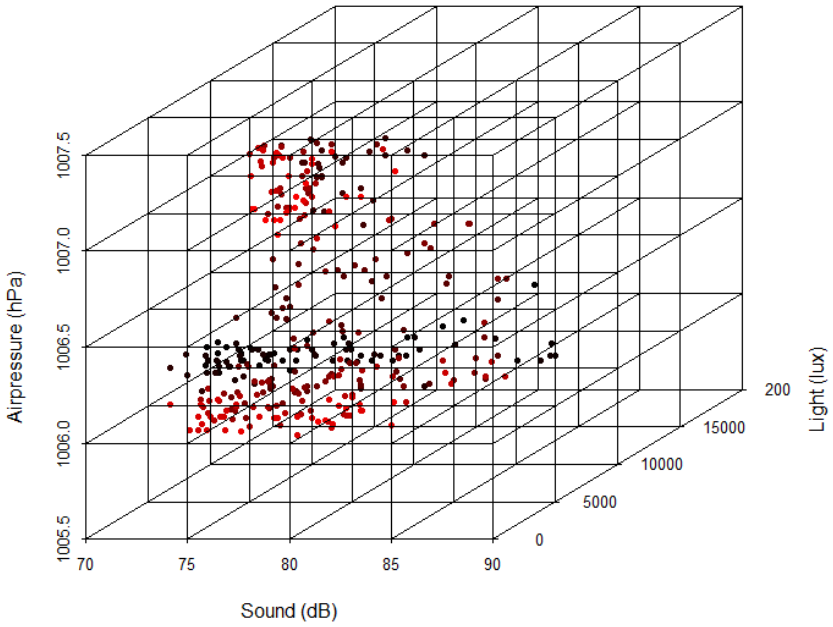


Figure 4.3: Multi-dimensional data space divided by cubes

4.3.5 Accommodating Multi-dimensional data

By now, *REPSense* method focuses on single dimensional data. However, we can easily generalize *REPSense* to accommodate multi-dimensional data. let ξ denotes a vector, then a υ-dimensional space is:

$$\xi^{\upsilon} = \{(\xi^{(1)}, \xi^{(2)}, \cdots, \xi^{(\upsilon)}) \mid \xi^{(i)} \in R\} \tag{4.10}$$

In the previous section 4.3.3, we divide single dimensional data ξ by div. In view of a general form, we also can divide multi-dimensional data ξ^{υ} by υ-cube. Therefore, single dimension is a special case in which υ-cube is a line, and in the case of 2 dimensions υ-cube is a rectangle. In addition, υ-cube is determined by each single $div^{(i)}$ which can be obtained from equation (4.7).

$$\upsilon - cube = div = (div^{(1)}, div^{(2)}, \cdots, div^{(\upsilon)}) \tag{4.11}$$

Figure 4.3 shows an example of 3 dimensional data space which consists of

sound, light and air pressure (atmospheric pressure) collected in our *trajectory sensing* system. Where v-cube is (5, 5000, 0.5), which means $div^{(1)}$ is 5 dB for $\xi^{(1)}$ (sound), $div^{(2)}$ is 5000 lux for $\xi^{(2)}$ (light), $div^{(3)}$ is 0.5 hPa for $\xi^{(3)}$ (air pressure). The number of v-cubes - m in a group is:

$$m = \prod_{i=1}^{v} (1 + \lceil \frac{max(G^{(i)}) - min(G^{(i)})}{div^{(i)}} \rceil), \quad where\ G^{(i)} \subset \xi^{(i)} \tag{4.12}$$

Finally, we can apply equation (4.4),(4.5),(4.6) without any modification to implement *REPSense* for multi-dimensional data.

4.4 Experimental Evaluation and Discussion

All experiments are performed by using real data set, which was collected in our trajectory sensing project in 4 months by 12 users. In every instance of data collection, sensing lasted from 30 minutes to 2 hours. Sampling period is set to 0.5 second, 1 second or 5 seconds in different instances but constant in a single data collection. The file size of sensor data from a data collection instance ranges from 600 KB to 5 MB.

4.4.1 Evaluation Criteria

We take advantage of sensor data collected in our project which include *ambient noise* and *ambient light* to evaluate our method. By drawing the histogram of our collected data, we roughly find the distribution of ambient noise is close to *Gaussian distribution* while the distribution of ambient light is close to *exponential distribution*. Using two different types of distribution can validate that our method is adaptable in multiple distributions. Our objective is to evaluate the accuracy over *compression ratio*. Here, *compression ratio (CR)* is defined as the number of points after reduction divided by the

number of original points. Since the reduced data meet the rule of equation (4.1) and (4.2), we can evaluate accuracy only by comparing data divergence of distribution, i.e., the similarity between original data's histogram and that of the reduced data. In probability theory, *f divergence* is used to measure the deference between two probability distributions. Let P and Q be two probability distributions over a space Ω such that P is absolutely continuous with respect to Q. For a *convex function* f such that $f(1) = 0$, *f-divergence* of Q from P is:

$$D_f(P \parallel Q) = \int_\Omega f(\frac{\mathrm{d}P}{\mathrm{d}Q})\mathrm{d}Q \tag{4.13}$$

There are many types of *f-divergence*; we select *Kullback-Leibler divergence*, *Hellinger distance* and *Itakura-Saito distance*, whose discrete forms are defined as equation (4.14), (4.15) and (4.16) respectively. They all are of non-negativity and non-symmetrical, but among them, *Itakura-Saito distance* is favorable because its convex function is consistent with our initial definition of *data diversity*.

$$\begin{gathered} D_{KL}(P \parallel Q) = \sum P_i \log(\tfrac{P_i}{Q_i}), \\ convex\ function\ f = t * log(t) \end{gathered} \tag{4.14}$$

$$\begin{gathered} D_H(P \parallel Q) = \tfrac{1}{\sqrt{2}}(\sum(\sqrt{P_i} - \sqrt{Q_i})^2)^{\frac{1}{2}}, \\ convex\ function\ f = (\sqrt{t} - 1)^2 \end{gathered} \tag{4.15}$$

$$\begin{gathered} D_{IS}(P \parallel Q) = \sum(\tfrac{P_i}{Q_i} - \log \tfrac{P_i}{Q_i} - 1), \\ convex\ function\ f = -\log(t) \end{gathered} \tag{4.16}$$

Aside from data divergence, we also evaluate our method in terms of data clustering performance. Moreover, we apply our method on a practical application to assess the real benefits.

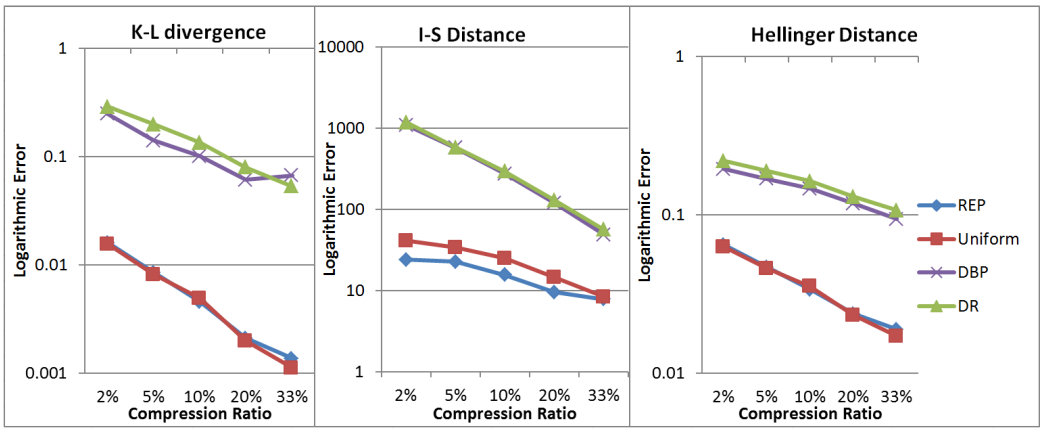


Figure 4.4: Divergence for ambient light over compression ratio: (a)*K-L divergence*, (b)*I-S distance*, (c)*Hellinger distance*.

4.4.2 Data Divergence Performance

In this case, experiments are conducted over single dimension data (ambient light and ambient noise respectively) in on-line mode (on smartphone before transmitting data). *K-L divergence*, *I-S distance* and *Hellinger distance* are employed to evaluate the accuracy over different compression ratio (2%,5%,10%,20% and 33%). Figure 4.4 and 4.5 show the result for ambient light and ambient noise respectively. We find our method performs best in terms of *I-S distance* (the smaller the value is, the better the performance is), and the advantage is more remarkable while under high compression ratio. *DR* seems incompetent because it drastically destroys *data diversity*. *DBP* performs almost same as *DR*. The reason is that *DBP* is a variant of *DR* plus a linear prediction model. Although *uniform sampling* is roughly the same as our method in terms of *K-L divergence* and *Hellinger distance*, it heavily depends on the consistency between sampling rate and *autocorrelation (ACF)*. Furthermore, because we focus on *data diversity* whose definition is similar to *I-S distance* that considers sum of information loss rather than average loss like *K-L divergence*, *I-S distance* is preferred in this work.

We also study the effect of input parameters on accuracy. From figure

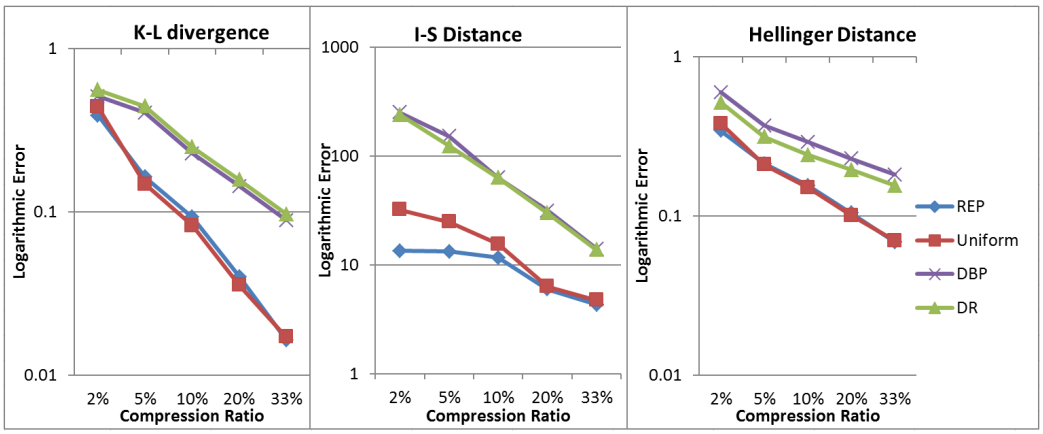


Figure 4.5: Divergence for ambient noise over compression ratio: (a)*K-L divergence*, (b)*I-S distance*, (c)*Hellinger distance*.

4.6, we can find: (1) given same value of w (w is the weight assigned to minority points in equation (4.5)), accuracy increases with compression ratio in general, (2) under given compression ratio, accuracy fluctuates with w. Given that our objective is to minimize the information divergence defined by equation (4.13), the process of optimizing w can be described as:

$$\arg \min_{w \in [0,1]} D_f(P \parallel Q) \tag{4.17}$$

In off-line mode we can calculate the entropy of original data ($H(P)$) since we have the posterior distribution of original data. We can generate Q (reduced data) which reaches the minimal of $L(Q)$ by sampling w, then based on equation (4.9), it will meet equation (4.17) in terms of *K-L divergence*. However, in on-line mode it is hard to determine the optimal value of w as we do not have the prior knowledge about distribution of original data. In addition, for a certain reasons, it is unnecessary to use an optimal w. For example, if more outliers are expected to be kept, w should be raised although divergence will be widened. In our experiments, we choose w from 0.05 to 0.2 according to empirical observation.

Another important parameter is buffer size which is the number of points

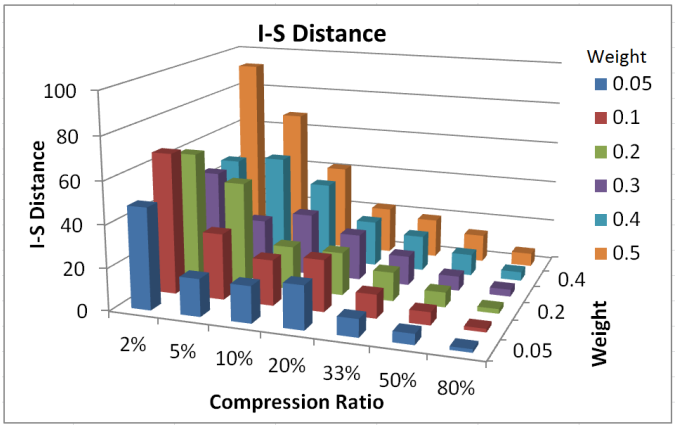


Figure 4.6: Weight (w) effects on *I-S distance* over *CR*

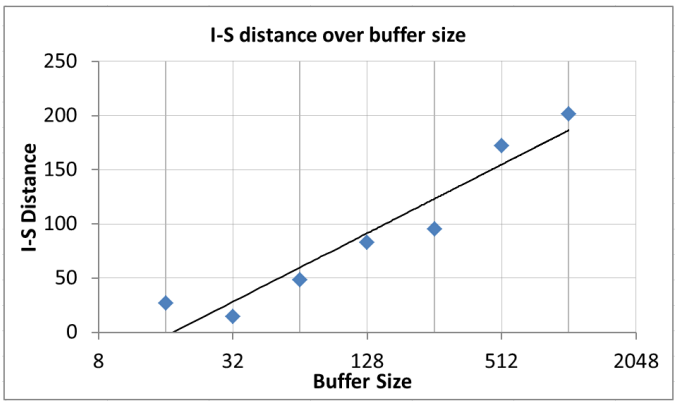


Figure 4.7: *I-S distance* over buffer size

in a group. Figure 4.7 shows inaccuracy over buffer size under the condition of *compression ratio* = 10% and $w = 0.3$. We find inaccuracy increases when buffer size increase, but the result can be improved by adjusting the value of w. In addition, as stated in section 4.3.4, there are two alternatives - *sum of information content* or *MBR area* - to calculate the magnitude of group in the first step of REPSense. Based on *Student t-test*, we find there is no significant difference in terms of accuracy between them, but MBR area is inexpensive in light of computation.

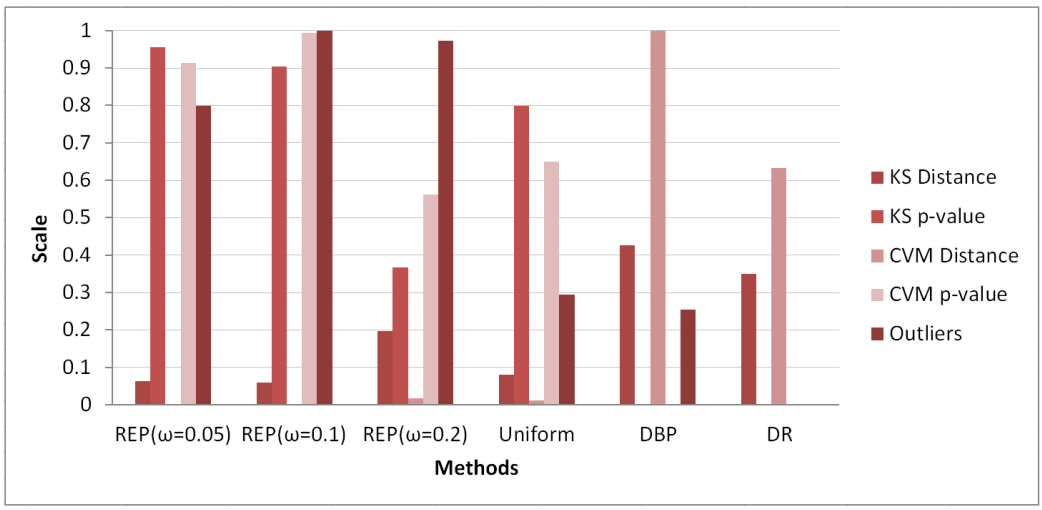


Figure 4.8: Goodness test and outliers detection

Off-line Data Reduction

In this case, we execute experiments in off-line mode (at the back-end server). Hence, our method adopts a global buffer whose size is the length of the entire file. Goodness of fit tests are adopted to evaluate performance over fixed $CR(10\%)$. Two well-known tests are used here, i.e. *Kolmogorov-Smirnov test (KS)* (see equation (4.18)) and *Cramér-von Mises test (CVM)* (see equation (4.19)).

$$D_{KS}(P \parallel Q) = \sup |Q(x) - P(x)| \tag{4.18}$$

$$D_{CVM}(P \parallel Q) = \int_{-\infty}^{+\infty} (Q(x) - P(x))^2 \mathrm{d}x \tag{4.19}$$

Where $Q(x)$ and $P(x)$ stand for the *EDF (Empirical Distribution Function)* of original data and reduced data respectively. *KS* test measures the widest distance between two distributions while *CVM* test measures the cumulative distance. In addition, a simple outlier identifying method is used to evaluate.

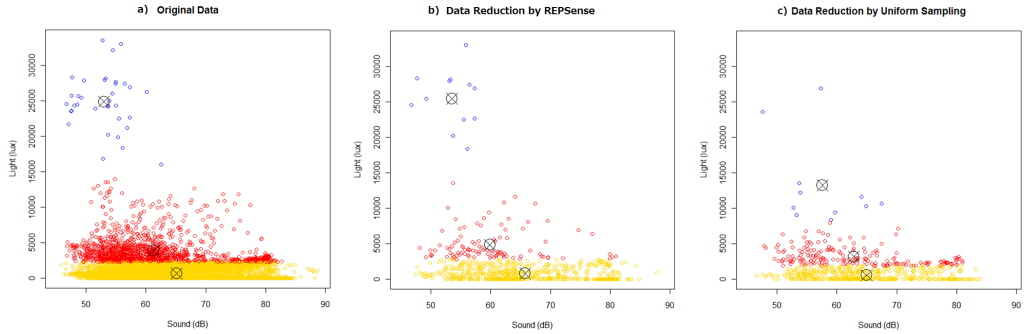


Figure 4.9: Clustering: a) original data, b) reduced data by *REPSense*, c) reduced data by *uniform sampling*

We define an outlier to be any observation outside the range:

$$\begin{gathered}\left[Q_1 - \lambda(Q_3 - Q_1), Q_3 + \lambda(Q_3 - Q_1)\right] \\ \textit{where } Q_1 \textit{ and } Q_3 \textit{ are the } 25^{th} \textit{ and } 75^{th} \textit{ percentile} \\ \textit{respectively, and } \lambda = 1.5\end{gathered} \tag{4.20}$$

The result is shown in figure 4.8, in which *KS* distance and the number of outliers are scaled to $[0, 1]$ for normalization. We claim that our method is better since it has short distance and high p-value in terms of both *KS* test and *CVM* test. In addition, as weight w increases, the number of outliers identified in our method increases and goodness of fit deteriorates as well. Probably it is because w raises the probability of outliers to be selected out while destroy the diversity balance simultaneously.

4.4.3 Data Clustering Performance

In this case, experiments are conducted over 2-dimensional data (ambient light and ambient noise) by comparing with *uniform sampling*. In addition, we measure the clustering performance in this case. *K-means* is used to classify original data and reduced data. To determine the number of clusters

(k^*) in a data set, which is a common problem, we simply adopt such a rule:

$$
\begin{aligned}
k^* &= \min\{k \mid \tfrac{totss_k - \sum_{i=1}^{k} withinss_k}{totss_k} \geq 0.9\} \\
where\ totss_k &= \tfrac{1}{{}_N C_2} \sum_{i=1}^{N} \sum_{j=i+1}^{N} (\parallel q_i - q_j \parallel_2)^2 \\
withinss_k &= \tfrac{1}{{}_{N_k} C_2} \sum_{i=1}^{N_k} \sum_{j=i+1}^{N_k} (\parallel q_i - q_j \parallel_2)^2 \\
N_k\ &is\ the\ number\ of\ points\ in\ k_{th}\ cluster,\ N = \textstyle\sum N_k \\
q_i\ &and\ q_j\ are\ single\ data\ points.
\end{aligned}
\tag{4.21}
$$

In our experiments, k^* ranges from 2 to 5 in the original data. Then, we use the same k^* when classifying reduced data by both *REPSense* and *uniform sampling*. As seen in figure 4.9, in our method the blue cluster holds more points than *uniform sampling* for our method places a higher priority on minority points. Subsequently, the center of blue cluster in our method keeps almost the same place as does in the original data while in *uniform sampling* it deviates much from the original data. In a word, we intuitively find that our method *REPSense* can generate more similar clusters comparing with original data than *uniform sampling*. Furthermore, we define the following 2 metrics to measure clustering performance.

- Cosine Similarity of Cluster Size: After clustering, we obtain an array of vectors $C_{S\{m\}} = (N_{c_1}, N_{c_2}, \cdots, N_{c_k})$, where $m = \{Rep, Unif, Orig\}$ stand for *REPSense*, *uniform smapling* and original data respectively. Here, N_{c_i} is the number of points in i^{th} cluster. Accordingly, cosine similarity of $C_{S\{Rep,Orig\}}$ and $C_{S\{Unif,Orig\}}$ are calculated.

- Average Distance of Cluster Center: We also obtain the center of each cluster by each method as follows: $C_{c\{m\}} = \{C_1(x_1, y_1), C_2(x_2, y_2), \cdots, C_k(x_k, y_k)\}$. Similarly, we calculate the *Euclidean* distance of each paired cluster for $C_{c\{Rep,Orig\}}$ and $C_{c\{Unif,Orig\}}$ after scaling the value of center to $[0, 1]$.

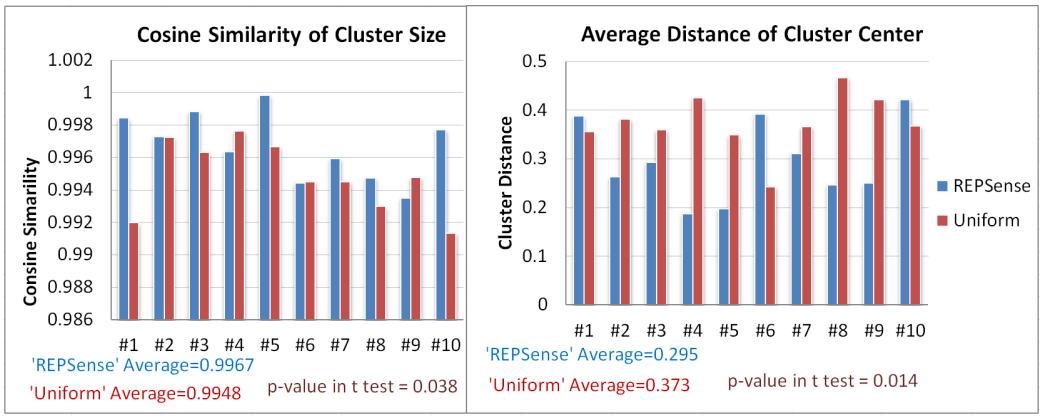


Figure 4.10: Clustering performance

Figure 4.10 shows the result of clustering performance, proving that our method fully outperforms *uniform sampling* in terms of above 2 metrics because it gets higher cosine similarity of cluster size and shorter average distance of cluster center. This is substantiated by *t-test* too. Note that the performance of our method highly depends on the value of w, however we can adjust w freely. Incidentally, expected running time of *K-means* algorithm is bounded by $O(n^{dk+1} \log n)$ [IKI94] where n is the number of data points and d is the dimensionality of data set. We can see the time cost will dramatically decrease if the number of points is reduced.

4.4.4 Target Application

Ambient light and noise at night are the energy emitted from the area through human activities. Thus, we claim the ambient light and noise can indicate how busy an area's night-life is. We also find that the ambient light and noise are spatially independent by hypothesis test using *K functions*, which imply that we can employ both of them to estimate how busy an area is. In our project, we collected ambient light and noise data around (within 500 meters) tens of train stations at night. In this application (it is also elaborated in section 6.1), we aim to calculate the **similarity matrix** among these train

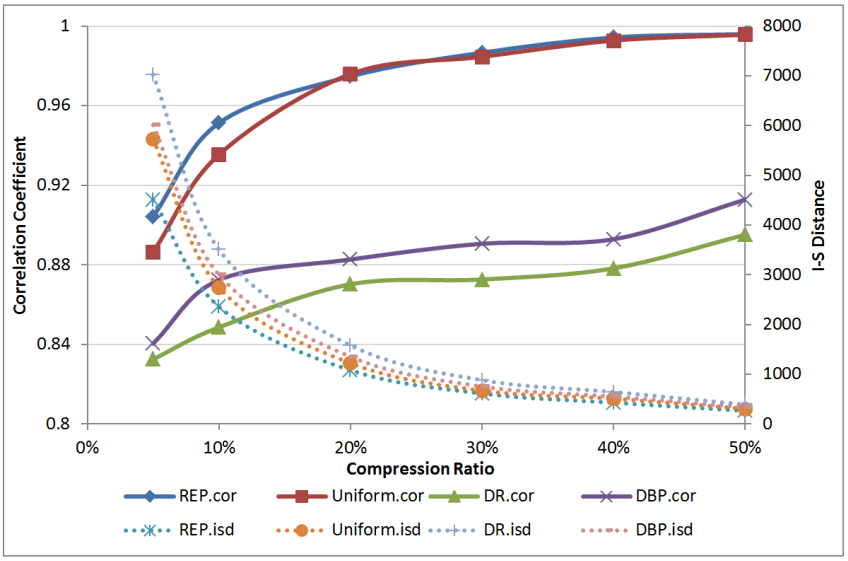


Figure 4.11: Correlation coefficient of similarity matrix and I-S distance

stations in terms of *Jensen-Shannon Divergence (JSD)*. The result is helpful to assess the value of business zone (around station).

We can obtain the light-noise joint distribution of every station and then calculate the *JSD* between each pair of stations. Assuming the number of stations M, to build the similarity matrix, the computation of *JSD* is $M * (M-1)/2$ times. Thus, it is significant to reduce data volume in light of computation cost. Now we apply our method to reduce data. After data reduction, we obtain the **similarity matrix** of stations again and compare it with the original similarity matrix by correlation coefficient. Figure 4.11 and figure 4.12 show the experimental results. In figure 4.11, the dot lines show the *I-S distance* and the solid lines show the correlation coefficient for our method and baseline methods. From it we can find: (1) our method has obvious advantage in both I-S distance (the lowest) and correlation coefficient (the highest), (2) the correlation coefficient is negatively correlated with *I-S distance*, which substantiates that using data divergence to measure accuracy is reasonable, (3) the correlation coefficient decreases with increasing compression ratio, which indicates that compressing too much will invali-

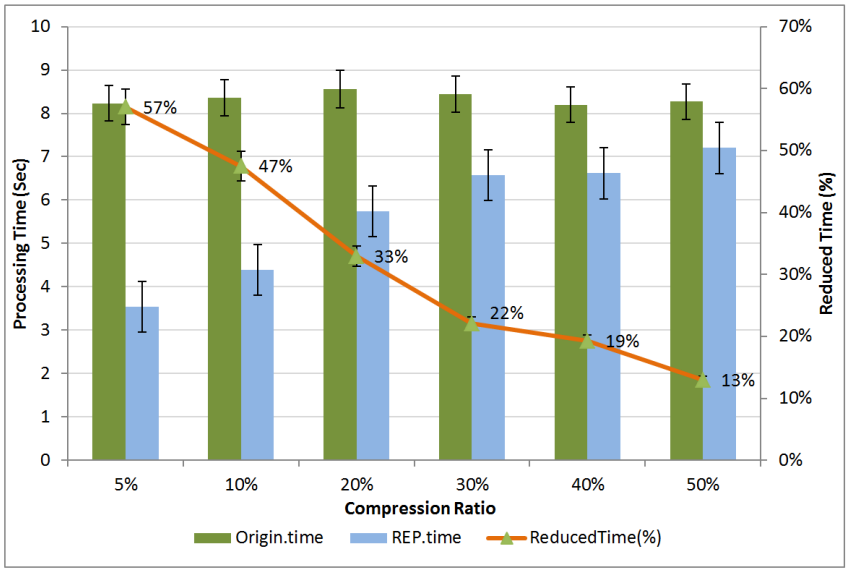


Figure 4.12: Processing time comparison

date the result. In figure 4.12, the processing time of generating **similarity matrix** is shown. As expected, the processing time after data reduction is linearly decreasing in proportion to compression ratio (the solid line shows the percentage of reduced time).

4.5 Chapter Conclusion

It is fairly challenging and significant to extract more useful data but reduce redundant data. Motivated by this, we propose a data model (*REPSense*) to represent data from the viewpoint of *data diversity*. We rethink data reduction problem by analogy to electoral system. In our method, data is selected by a fashion which is similar to electing congressional representatives. We also design a novel scheme: *divide/merge principle* and *selection strategy* which implement *REPSense* model to reduce data. Although we don't theoretically prove our model is effective, we draw our conclusion based on a large amount of experiments with real data set. Through comparing with conventional methods and state-of-the-art methods, our method performs well in

terms of data divergence especially *Itakura-Saito distance*. Our method also outperforms other methods in terms of data clustering and practical data analysis application. It is foreseeable that data reduction will linearly mitigate storing and transmitting problems, though we did not evaluate it by experiments.

In future we plan to explore optimal parameters of our method, and evaluate it from other viewpoints, e.g., transmitting cost, outlier detection performance.

Chapter 5

Sensor Application: iBaro-altimeter

5.1 Introduction

Localizing the position of a moving target is the basis of all location based services (LBS). Until now researchers have mainly focused on localization on a 2D plane (latitude and longitude). However,altitude or elevation is also important for LBS including 3D navigation, indoor localization (e.g. floor), mountaineering, emergency rescue operations and so on. Although GPS can provide altitude,its accuracy is notoriously poor and unstable (sometimes the error can reach up to 2.5 times more than the horizontal one) [KH06]. Moreover, it does not work in indoor environments. There are other means to measure height such as hypsometer, ultrasound and radar based techniques. However, they can only provide relative height (not elevation). Moreover they are not readily available due to cost and limitation on their scope of applications. Therefore, the objective of this research work is to develop a ubiquitous, less costly and highly accurate elevation measurement system.

Our method is based on a well established theory that air pressure (atmospheric pressure) change corresponds to elevation changes. Air pressure

can be measured using a barometer. Based on this law, altimeters used in aircrafts use barometer to get the air pressure and use this information to calculate the altitude. Moreover, they need local air pressure information from control towers as reference points. These reference points can be used to calculate the elevation with acceptable accuracy only when one is within certain distance from that reference point.

Thanks to the development of sensor technologies, air pressure sensors are built-in in most smartphones nowadays, which creates new opportunity to take advantage of the baro-altimeter technology. However, we have to tackle certain challenges before using these low cost embedded barometers to get accurate measurements for elevation. In a standalone mode (elevation measured using smartphone's barometer only,based on ISA model [NAS]), the error reaches tens of meters or even hundreds of meters (depending on weather conditions and altitude). Even other research works [LHG13] using meteorological stations as reference points (as used by aircrafts), reported errors in the range of several meters to tens of meters. In this work we propose a system called iBaro-altimeter that adresses this shortfall by solving three sub-problems.

The first problem we address is related to the density of reference points. As mentioned earlier, accuracy of the measurement of elevation at a target depends on the distance between this point and the reference point. Therefore, we need more reference points to make sure that smartphones are always within acceptable range from at least one reference point. However, the only reference points available are meteorological stations which are often sparsely located. Hence, we propose methods to introduce ad-hoc reference points.

The second problem arises from the fact that these meteorological stations broadcast periodically, usually at one hour interval. However, within this period, the air pressure might change substantially. We integrate information from multiple reference points including the ad-hoc ones and also use a forecast model to estimate the air pressure on demand.

The third problem we tackle is related to the accuracy of the barometer sensors embedded in smarphones. Earlier research work [KWS12] showed that these sensors are less accurate and less stable than the barometers certified for professional use. We use multiple filtering techniques such as Kalman filter to handle this issue.

We believe that the ability to obtain elevation on smartphones would be beneficial in various potential applications, for example:

- *Tsunami evacuation*:
  In case of a Tsunami, it is advisable to run to higher ground in order to survive. However, the problem is that it is not easy for most people to know which way/direction is higher. Even availability of evacuation maps doesn't help much because they are difficult to use and they also provide height of the ground and not the actual elevation(e.g of a person in a building). Easy application of our method would enable a smartphone to provide accurate elevation and with ease of use this would save a lot of lives in an event of a Tsunami.
- *Car navigation*:
  On flyover/overpass roads, GPS of car navigation cannot localize which road (underpass or overpass) the car is on, especially if it is at the beginning of navigation process. This can lead to wrong or sub-optimal routing because obviously the route through the underpass or overpass may be different. Availability of accurate elevation of a user would easily solve this problem and eanable the navigation system to distinguish between an overpass and underpass road segment.
- *Mountain climbing*:
  For mountain climbers, altimeter is one of indispensable gears. To record or measure ones' elevation is crucial for safety as well as enjoyment. Though altimeter watches are available, smartphone based altimeters would provide an attractive alternative for some users depending on cost, ease of use and accuracy.

- *Floor based services*:
  Our system can easily identify on which floor a person is by estimating relative height. This information would save lives, in an emergency rescue operation inside buildings as survivors' floor can be easily identified. Numerous other applications for instance, floor based games, social network (friend finder), indoor positioning would also benefit from this data.

  Our main contributions are as follows:

  - We proposed an integrated framework to provide accurate elevation measurement which uses only smartphones without special infrastructure by
    - employing multiple techniques to handle systematic error and random error;
    - integrating heterogeneous reference points;
    - applying adaptive extended Kalman filter to smoothen elevation profile;
    - leveraging short-term forecast model to enhance the accuracy and
    - estimating the error interval of obtained elevation on-line.
  - We evaluated the system in different indoor and outdoor settings.
    - Case 1 (Outdoor Walking): Participants carrying smartphones with our developed tool installed on it,walked randomly in an urban area.
    - Case 2 (Mountain climbing): Experiment participants climbed 3 mountains (elevation ranges from 600 meters to 1900 meters).
    - Case 3 (Inside buildings): Participants walked, took stairs, took elevator in 3 buildings.

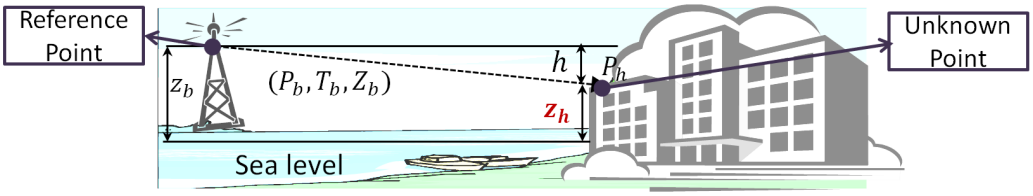


Figure 5.1: The theory of elevation measurement from air pressure

5.2 iBaro-altimeter Overview

5.2.1 Theory of elevation measurement from air pressure

The relation between air pressure and elevation has been well studied, and it is defined as:

$$P_h = P_b \cdot \left[\frac{T_b}{T_b + L \cdot (Z_h - Z_b)}\right]^{\left[\frac{g \cdot M}{R^* \cdot L}\right]} \tag{5.1}$$

where

Z_h, Z_b: height above sea level (meter)
P_h: air pressure (pascal) at height Z_h
P_b: air pressure (pascal) at height Z_b
T_b: standard air temperature (K)
L: temperature lapse rate (K/m)
g: gravitational acceleration (m/s^2)
M: molar mass of earth's air $(kg/kmol)$
R^*: universal gas constant for air $(N * m/(kmol * K))$

It can be derived easily using the ideal gas law with the assumption that all pressure is hydrostatic(for details see [NAS]).

To calculate elevation, we can inverse equation 5.1 as:

$$Z_h = Z_b + \frac{T_b}{L}\left[\left(\frac{P_h}{P_b}\right)^{-\frac{L \cdot R}{g}} - 1\right] \tag{5.2}$$

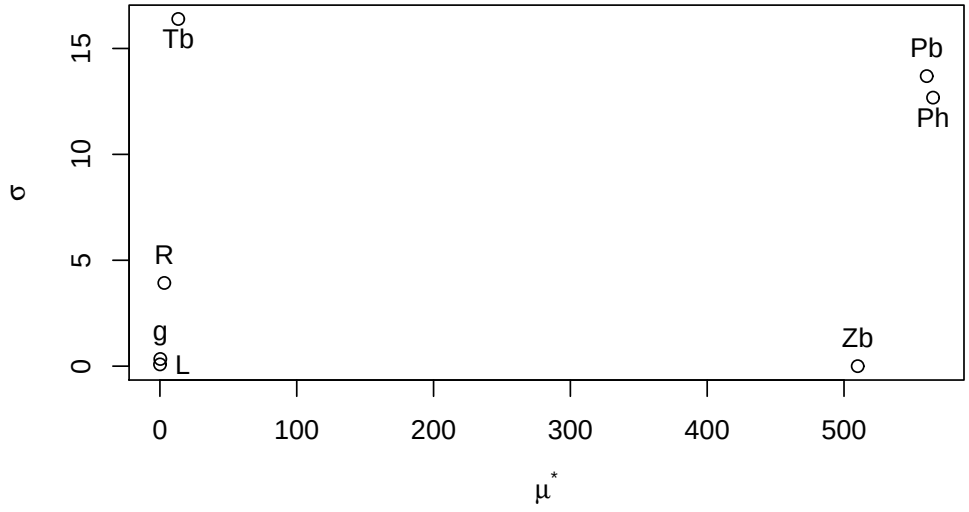


Figure 5.2: Sensitivity of input parameters of baro-altimeter equation

where $R = \frac{R^*}{M}$.

As shown in figure 5.1, if status (P_b, T_b, Z_b) at a reference point is known and air pressure (P_h) is known too, it is easy to obtain the elevation (Z_h) from equation 5.2 and relative height (by $h = Z_h - Z_b$).

5.2.2 Sensitivity analysis

The challenge in elevation measurement is that there are always errors due to inaccurate measuremtns associated with each of the input parameters in equation 5.2. Therefore we applied elementary effects method (of sensitivity analysis) to simulate the influence of parameters on the result. Elementary effects (EE) method [Mor91] can identify non-influential inputs and rank them in order of importance. EE of Compolongo [CCS07] provides two sensitivity measures for each input parameter:

- μ^*, assesses the overall importance of an input on the model output.

Table 5.1: Elevation variation caused by inputs' error

		$\Delta Z_h(m)$ caused by			
Zh^* (m)	Tb(°)	$\Delta P_h[1hP]$	$\Delta P_b[1hP]$	$\Delta Z_b[1m]$	$\Delta T_b[1°]$
100	0	-7.96	7.87	1	0.32
100	15	-8.40	8.31	1	0.32
100	30	-8.84	8.74	1	0.32
500	15	-8.84	8.74	1	0.34
1000	15	-9.33	9.21	1	0.36
2000	15	-10.49	10.34	1	0.40

Z_h^*: around this height; $1hP = 100Pascals$

- σ, describes non-linear effects and interactions.

Low values of both $\mu*$ and σ correspond to a non-influent input. Figure 5.2 shows the result, which suggest that P_b and P_h are the most influential inputs, Z_b is also important, while g, L and R are the least influential inputs, T_b is less influential but affects the variability substantially. Consequently, in order to enhance accuracy of elevation 5.2, we focus on reducing uncertainty of P_h, P_b, Z_b, T_b. Although g will vary with latitude, L is subject to elevation, and R depends on the humidity of air, we set them as constants by following the international standard atmosphere model [NAS].

In addition, we made a numerical results table for effects of influential inputs (table 5.1) based on the concept of the total differential [Ste95]. First, we linearise the total differential of equation 5.2 by taking partial derivative on every input. Then, moving one input at a time while keeping other inputs at their baseline values to get output changes. From the table, we can verify the conclusion of EE.

5.2.3 iBaro-altimeter architecture

As explained in the previous section, accurate reference point information is crucial for obtaining highly accurate elevation when using the Barometric based method. Since there are meteorological stations for weather forecast service in most countries, it is a common practice to them as reference points.

However, these stations are sparsely located and they broadcast the information once in every hour or even once in every 3 hours. For example, in Japan there are only 156 stations which measure air pressure, and they broadcast information every hour. This coarse spatial density (about tens of kilo-meters) and low update frequency (about one hour) results in lower accuracy when elevation is obtained from barometer.

Here we propose a new system called *iBaro-altimeter* to resolve this problem. In our system we incorporate three types of reference points.

- *Base station*:
  These are meteorological stations usually provided by national meteorological agency.
- *DEM reference*:
  It is based on Digital Elevation Model (DEM). When a person (with smart phone) is on the ground in outdoor, it is possible to get the elevation from DEM map based on his current location (from GPS). When combined with air pressure measurement from the same smartphone, this status can be used as a temporal reference point. Every smartphone only maintains one DEM reference point or none at all. This reference point stores the latest elevation when GPS signal is good enough.
- *Smartphone reference*:
  It is based on a smartphone within our system. If elevation of the smartphone is obtained from our system, then this smartphone itself can be used as an ad-hoc reference point. To avoid error accumulation, the *smartphone reference* cannot be refereed to itself (only can be used by other smartphones).

Figure 5.3 shows the architecture of *iBaro-altimeter*. To calculate elevation of smartphone X, first it needs to get GPS information (latitude, longitude, altitude, accuracy). Then, using this information it requests the reference points server. On the other end, the server will integrate all refer-

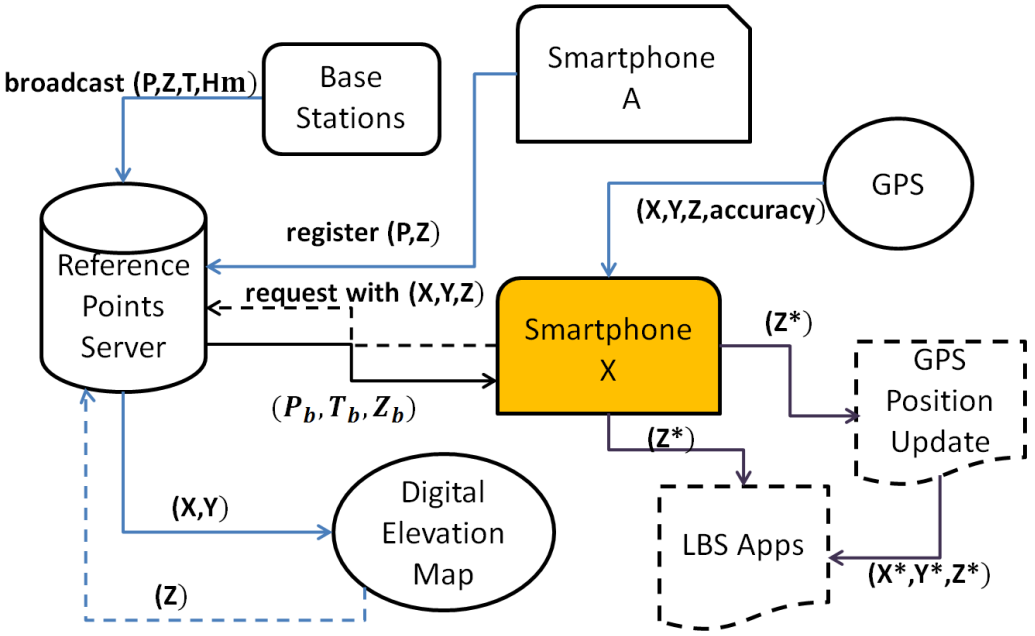


Figure 5.3: iBaro-altimeter architecture (X:longitude, Y:latitude, Z:altitude(elevation), P: air pressure, T: temperature)

ence points within a specified spatio-temporal radius around the smartphone and send the resulting reference point back to the smartphone. Finally, based on this virtual reference point, the smartphone can obtain accurate elevation which can be used in other LBS applications. Furthermore,it can also be used to improve GPS accuracy. Figure 5.4 shows how *iBaro-altimeter* works in a client/server fashion while details of major components are provided in the next section.

5.3 Components of iBaro-altimeter

5.3.1 Sensor calibration

Barometers built in smartphones are not as accurate as the measurement devices of meteorological stations. There is a systematic error between air

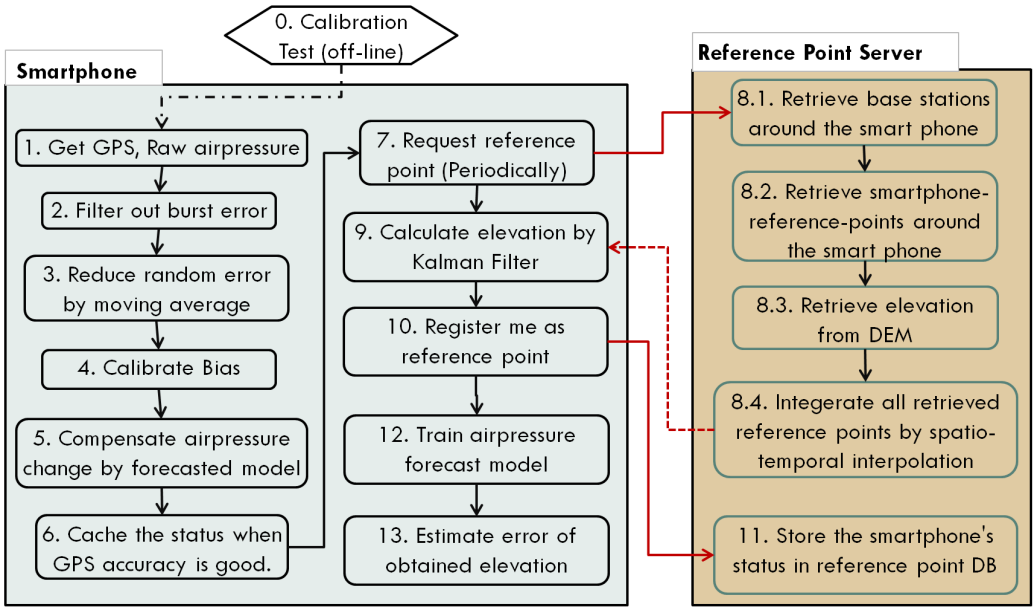


Figure 5.4: Workflow of iBaro-altimeter

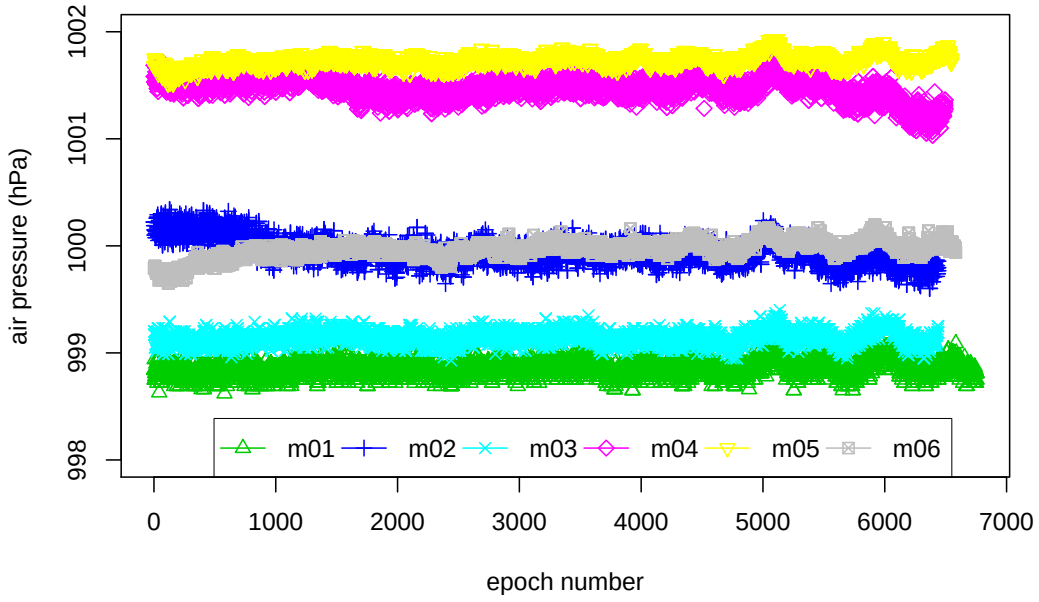


Figure 5.5: Reading differences among pressure sensors

pressure value read from barometer of smartphone and the actual air pressure. According to the specification of smartphone's barometers, there is a maximum absolute error of 4 hPa. We do calibration test to eliminate this systematic error before using the smartphone's barometer. Figure 5.5 shows the pressure sensor readings in 6 smart phones at identical location and time. Among these 6 smartphones, there are three different models of barometer (type 1: $m01$ and $m03$, type 2: $m02$ and $m04$, type 3: $m05$ and $m06$). We found that the systematic error is different, even independent on models, but it is stable over longer period of time. Calibration test can be done when the elevation Z_h and Z_b, T_b, P_b are known exactly. According to equation 5.1, P_h can be obtained with known Z_h and Z_b, T_b, P_b. Then the systematic error is the difference between the obtained P_h and the actual reading on barometer. By repeating the test, standard deviation can be obtained for evaluating the stability of barometer. The calibrated bias will be added before using the raw value of barometers.

5.3.2 Filtering burst error and random error

As can be seen in figure 5.6 there are random errors and burst errors (spikes) in raw values of barometer (sampling rate: 2 Hz). According to the specifications of the barometer, there is a random error of less than ± 0.1 hPa (depends on models). It is easy to reduce this random error by a low pass filter or simply taking average.

On the other hand, burst error which is caused by electronic defect or drastic change in air pressure, can be identified by a threshold-based method. Suppose the air pressure at time t is P_t , then we detect burst error by this condition:

$$|P_{t+1} - P_t| > \Delta = e_r + (e_h + 1.96 * e_w) * \Delta T \tag{5.3}$$

where e_r is random error (about 0.1 hPa), e_h is air pressure change caused

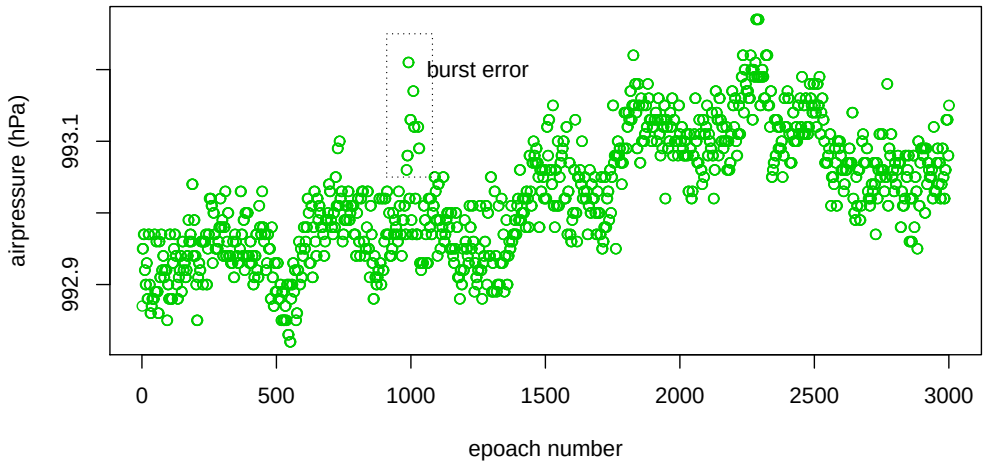


Figure 5.6: Random error and burst error of barometer

by elevation change, in normal situation we assume fast elevation change (by taking elevator) is $9m/s$ (about $0.1hPa/s$), e_w is air pressure variation (σ) in 1 second from weather change, which can be estimated from historical data of air pressure, ΔT is the sampling interval (second). This threshold based method might not completely remove all burst error (depends on how we define burst error), nevertheless burst error will be further addressed at a later stage with adaptive Kalman filter.

5.3.3 Integrating reference points

In our system, we introduce two types of dynamic reference points(i.e. *DEM reference* and *smartphone reference*), and fixed *base stations*. As shown in figure 5.7, *base stations* are stationary and broadcast information at fixed time interval, while *smartphone references* are mobile and update information dynamically, and *DEM reference* is available when GPS signal is very accurate. To estimate the elevation of $smartphoneX$, it is obviously not best to use the nearest or the latest reference point. Because (1) the nearest one

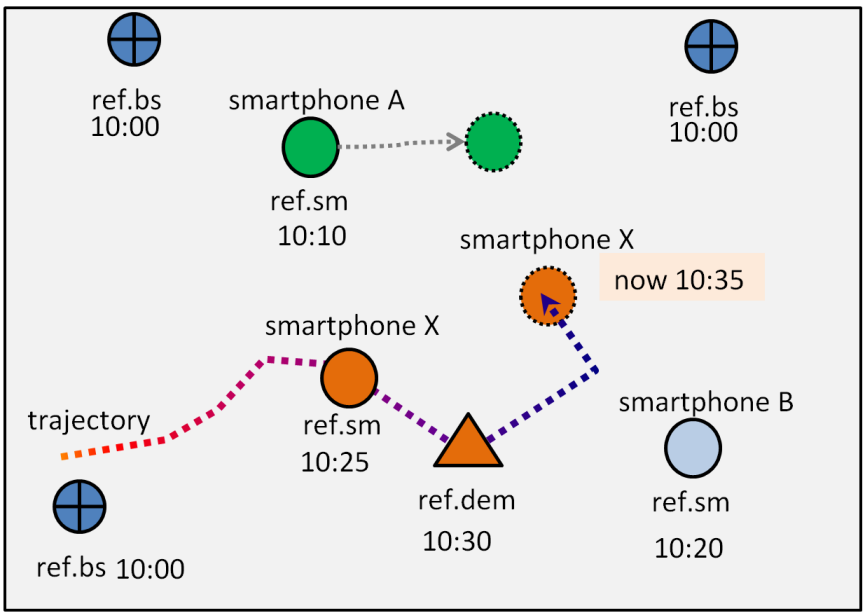


Figure 5.7: Reference points with dynamic spatio-temporal feature

may be not the latest one and vice versa, (2) all reference points are not fully accurate, meaning using a single point results in error accumulation. As such, we take into account all the reference points which are within a specified distance and time period (e.g. within 50 km and 1 hour until now). When a reference point is closer to the smartphone, the accuracy of elevation measurement is better, and the newer the reference information, the more accurate the measurement. In addition, the vertical distance between reference point and smartphone affects the accuracy too.

Each factor (horizontal distance, vertical distance and time gap) independently leads to error accumulation in a certain form (let's say $f(hs), g(vs), h(t)$ respectively). We assume their form follows an exponential function (Gaussian distribution) which can be verified by approximating the error distribution with regression method. Therefore, we propose a model to integrate all reference points to generate combined reference information. Each reference

point is given a weight w based on spatio-temporal distance as follows:

$$w = f(hs) * g(vs) * h(t) \tag{5.4}$$

where we assume $f(hs)$, $g(vs)$ and $h(t)$ follow $N(0, \sigma_{hs}^2)$, $N(0, \sigma_{vs}^2)$, $N(0, \sigma_t^2)$ respectively, then

$$w = (2\pi)^{-\frac{3}{2}}(\sigma_t \sigma_{hs} \sigma_{vs})^{-\frac{1}{2}} e^{-\frac{1}{2}[\frac{t^2}{\sigma_t^2} + \frac{hs^2}{\sigma_{hs}^2} + \frac{vs^2}{\sigma_{vs}^2}]} \tag{5.5}$$

where t is the time gap between the reference information time-stamp and current time, hs is the horizontal distance (calculated from latitude and longitude), vs is the vertical distance (altitude) between the reference point to the average elevation of base stations (see Z_b in algorithm 4). We also assume that the effect of a factor should be marginal,when a factor reaches a specific value. Therefore, all these 3 factors will be scaled to [0,1] by dividing by a specific value. For example, time gap t is divided by one hour ($MaxT$) , horizontal distance hs is divided by 30 Km ($MaxHS$), and vertical distance (vs) is divided by 500 meters ($MaxVS$). Therefore the weight becomes very small (i.e. 5 %) when time gap is greater than one hour, and same for other 2 factors. Note that, all these 3 factors can be scaled to [0,1] just because we search for reference points only within a specific time period ($MaxT$) and a specific distance ($MaxHS$ and $MaxVS$).

In addition, although these 3 factors are considered independent, they are supposed to contribute to the weight at different rate of change. For example, one unit (say 1 minute) of time gap may not lead to same error as one unit (say 1 Km) of horizontal distance. This rate of change is related the standard deviation (σ^2), therefore we can control it by adjusting σ_t^2, σ_{hs}^2 and σ_{vs}^2. From history of air pressure in different meteorological stations, we can estimate how much the air pressure changes with respect to specific time interval and distance. Let's say, in $MaxT$ (1 hour) the change of air pressure is ΔP_t hPa, in $MaxHS$ (30 Km) it is ΔP_{hs} hPa, in $MaxVS$ (500

m) it is ΔP_{vs} hPa. Note that ΔP_{vs} is not the change of air pressure due to elevation but the error caused by conversion in barometric calculation (i.e. equation 5.1). Variance $(\sigma_t^2, \sigma_{hs}^2, \sigma_{vs}^2)$ should be inversely proportional to $\Delta P_t, \Delta P_{hs}, \Delta P_{vs}$. If we fix one of the three, we can determine the other two according to the proportion. In our experiment, we set the weight for time gap: $P(t \geq 1hour) = 5\%$, which means $1.96\sigma_t = t = 1$. And σ_{hs} is calculated by: $\sigma_{hs} = \sigma_t * \Delta P_t / \Delta P_{hs}$, σ_{vs} can be calculated in similar way.

In our system, heterogeneous reference points(static and dynamic) coexist. Different types of reference points have different accuracy due to their intrinsic characteristics. We need to assign different weight according to the type of reference points. Base stations usually perform good and are stable. We can give a high constant credit (*credit.bs*).

DEM reference has error due to the DEM map generation process, thus we give a constant credit *credit.dem* less than *credit.bs*. Moreover, DEM reference's credit also depends on the geographic feature where the target is located when retrieving it. If you are on the roof of building, then DEM reference has large error since DEM elevation does not consider any artificial constructions. Since GPS altitude is available and relatively reliable when DEM reference is created, we estimate the variable credit of DEM reference according to the difference between DEM elevation and GPS altitude, which is defined as:

$$cde = e^{(-\frac{|Z_{dem} - Z_{gps}|}{\lambda})} \tag{5.6}$$

where λ is a parameter based on empirical evaluation. Note that GPS altitude is based on reference ellipsoid while DEM elevation is based on geoid, but it is easy to convert ellipsoid coordinate to geoid coordinate.

Smartphone reference is not very reliable due to the cheap barometer's performance, thus we give a constant credit *credit.sm* less than *credit.dem*. Moreover, Smartphone reference's credit also depends on the situation when

its elevation is calculated. If the smartphone obtained the elevation from a lot of high reliable reference points, then the credit will be high. It means the credit is proportion to the sum of weight (w) when its elevation is calculated.

Algorithm 4 shows the detail of how to integrate all reference points. It returns a tuple of (P_b, T_b, Z_b) which is called reference information. In line 6, temperature of the obtained reference point is obtained by inverse distance weighting from base stations only. In fact we should consider the elevation adjusted temperature that is adjusted by temperature lapse rate with subject to elevation. Note that the weight w is normalized by its sum, thus its value is relative not absolute.

5.3.4 Short-term forecast of air pressure

Since reference points are not updated frequently and requesting the reference points server is expensive (in terms of both communication and computation), the smartphone only requests the server at specific time intervals. In addition, it is unnecessary to get reference information too often because weather conditions don't change rapidly. Moreover, it is reasonable to predict the air pressure in a short time period (less than 30 minutes). (For longer time prediction, we found the error is extremely large (3 hPa) after we verification with actual air pressure measurements and forecast data provided by Meso-scale weather forecast model.) Consequently, we compensate the air pressure which is caused by weather change within the interval of reference request.

As shown in figure 5.8, suppose the air pressure on smartphone is P_{sm}. We convert it to P_{local} on a fixed location (by equation 5.1) to remove the influence of elevation change. Hence, the air pressure P_{local} represents changes with respect to time due to weather change only. Suppose the latest reference point request is at time T_i, we will predict the air pressure change (ΔPw_k) between T_i and next request time T_{i+1}. The predicated air pressure changes will be used to compensate the air pressure reading on the smartphone, which

Algorithm 4 Integrating reference points

1: **function** IntegrateRefs(cur.time, cur.lat, cur.lon, dem.time, dem.lat, dem.lon, dem.alt, dem.P_s)
▷ *search base stations and smartphone references within given spatio-temporal range*
2: $ref\_bs = SearchBS(cur.lat, cur.lon.cur.time)$;
3: $ref\_sm = SearchSM(cur.lat, cur.lon.cur.time)$;
▷ *request elevation from DEM map*
4: $ref\_dem = RequestDEM(dem.lat, dem.lon)$;
5: $Z_b = Average(ref\_bs.Z_s)$;
▷ *get temperature by inverse distance weighting the temperature of base stations*
6: $T_b = IDW(ref\_bs.T_s)$;
7: $P_b = 0, w_{sum} = 0$;
8: **for all** ref_pt in $(ref\_bs, ref\_sm, ref\_dem)$ **do**
▷ *CalcPressure refers to equation 5.1*
9: $P_x = CalcPressure(T_b, ref\_pt.P_s, ref\_pt.Z_s, Z_b)$;
10: $hs = Distance(ref\_pt.lat, ref\_pt.lon, cur.lat\ , cur.lon)$;
11: $vs = ref\_pt.Z_s - Z_b$;
12: $t = ref\_pt.time - cur.time$;
13: $w = TVN(hs, t, vs)$; ▷ *TVN refers to equation 5.5*
14: **if** $ref\_pt \in ref\_bs$ **then**
15: $w = w * credit\_bs$;
16: **else if** $ref\_pt \in ref\_sm$ **then**
▷ *ref_sm.score is the normalized w_{sum} based on ref_sm's previous measurement.*
17: $w = w * credit\_sm * ref\_sm.score$;
18: **else if** $ref\_pt \in ref\_dem$ **then**
19: $w = w * CDE(dem.alt, ref\_pt.Z_s) * credit\_dem$; ▷ *CDE refers to equation 5.6*
20: **end if**
21: $P_b = P_b + P_x * w$;
22: $w_{sum} = w_{sum} + w$;
23: **end for**
24: $P_b = P_b / w_{sum}$;
25: **return** (P_b, T_b, Z_b);
26: **end function**

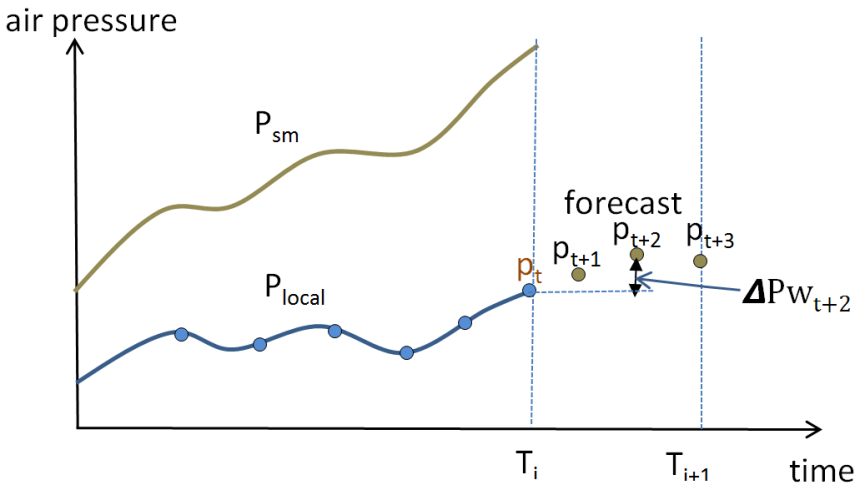


Figure 5.8: Illustration of air pressure forecast

could reduce the error due to the old reference information until next request.

Since we only predict the air pressure over a short time around a certain location, it is reasonable to deal with air pressure as time series data and ignore spatial variations. Although atmospheric pressure has daily and seasonal tides due to the Moon's gravitational pull and the Sun's heating, we don't consider these seasonal patterns because their effect is negligible over short time periods. There are many forecasting models for time series data, including ARIMA, Holt-Winters method, exponential trend method, damped trend method and so on [HA13]. We applied additive dampened trend method because it is more flexible (for details see [HA13]).

Incidentally, the error and variance of predicted air pressure can be estimated using the method proposed by R.J. HYNDMAN [HKOS05], which will be utilized in error estimation later.

5.3.5 Estimating elevation

Although it is workable to directly calculate the elevation (Z_h) from reference point's information (T_b, P_b, Z_b) and smartphone's barometer (P_h), the elevations of trajectory (called elevation profile) are not stable because smart-

phone's barometer is too sensitive to the environment. As such, the obtained elevation looks jerky, which does not follow the natural movement of users (human). To tackle with this problem, we apply an adaptive extended Kalman filter. Considering elevation measurement is a non-linear system, we don't know the true elevation, but can estimate it from the observation (air pressure). The states ($\mathbf{x}$) we are concerned with are elevation Z_h and elevation velocity $\dot{Z}_h$, the measurement ($\mathbf{z}$) is only air pressure (P_h). For process model, we assume it follows Newton's equation of motion:

$$\mathbf{x_{k+1}} = \begin{bmatrix} Z_{h_{k+1}} \\ \dot{Z}_{h_{k+1}} \end{bmatrix} = \mathbf{f}(\mathbf{x_k}) + \mathbf{w_k} = \begin{bmatrix} Z_{h_k} + T\dot{Z}_{h_k} + w_{1_k} \\ \dot{Z}_{h_k} + w_{2_k} \end{bmatrix} \tag{5.7}$$

where T is the time interval of every step and $\mathbf{w_k}$ represents the process noise. The following is the measurement model:

$$\mathbf{z_k} = \begin{bmatrix} P_{h_k} \end{bmatrix} = \mathbf{h}(\mathbf{x_k}) + \mathbf{v_k} = \begin{bmatrix} P_b \cdot [\frac{T_b}{T_b + L \cdot (Z_{h_k} - Z_b)}]^{[\frac{g \cdot M}{R^* \cdot L}]} + v_{1_k} \end{bmatrix} \tag{5.8}$$

where $\mathbf{v_k}$ represents the measurement noise.

As shown in equation 5.2, the measurement (air pressure) relationship to the process (elevation) is non-linear, thus we have to apply Extended Kalman Filter (EKF) in our system. EKF will recursively perform two steps: model forecast step and data assimilation step, to estimate the elevation Z_{h_k} at time k (for details see [WB95]). The Jacobian of $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are required in the model forecast step and data assimilation step respectively. In our model,

$$\mathbf{J_f} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \mathbf{J_h} = \begin{bmatrix} \frac{\partial P_h}{\partial Z_h} & 0 \end{bmatrix} \\ where\ \frac{\partial P_h}{\partial Z_h} = P_b * \frac{-g}{RT_b}(1 + \frac{L}{Tb}(Z_h - Z_b))^{-\frac{g}{LR}-1} \tag{5.9}$$

Kalman filter can estimate the elevation which reaches the minimum error $(E(e_k e_k^T) = (x_k^f - x_k^a)(x_k^f - x_k^a)^T)$.

In addition, we simplify the movement of human by Newton's equation of motion, although this model cannot fully match the movement (in elevation). Thus we apply an adaptive Kalman filter proposed as introduced in [MS99] to adjust the process noise.

5.3.6 Error estimation

Although we strived to reduce errors from various sources in our estimation of elevation, the result still contains some errors. Therefore, it is important to notify the users on-line of the error interval associated with every measurement. All possible error sources are classified as follows:

- Error from inaccurate reference points (e_{ref}):
  Reference points are assigned weights based on their spatio-temporal distance and intrinsic characteristic. Thus this error can be estimated from the sum of weights from all integrated reference points. We approximate the error relation from history data by regression method as: $e_{ref} = (\log(\frac{N_s}{ref.score}) + 1) * E_u$, where E_u is the average error (meters) when the sum of weights is greater than N_s (predefined score) from history data, and $ref.score$ is the sum of weights from all integrated reference points in current measurement.

- Error due to weather change ($e_{weather}$):
  Reference point information is requested periodically, and reference information will gradually become less accurate since weather (atmospheric pressure) changes with time and location change. If weather changes drastically, then the error of reference information will increase accordingly. Thus, the variance of weather change indicates magnitude of this error. On the other hand, our air pressure forecast model (section 5.3.4) generates the variance of predicted air pressure (σ_w). Therefore the error ($e_{weather}$) is

roughly determined by $e_{weather} = 1.96\sigma_w * E_p$, where E_p (about 9 meters) is the elevation change caused by 1 hPa pressure change .

- Error due to random error of barometer (e_{random}):
  e_{random} depends on the quality of barometer, which is set as a constant according to the product specification (usually 1 meter). Since we have reduced the error by taking average, this error should be less than what the specification claims.
- Error due to systematic error of barometer (e_{sys}):
  Just like random error, the systematic error depends on the quality of barometer. As we did calibration test before using it, the variance of calibrated bias can indicate the error e_{sys}. In addition, there is burst error in barometer, however we consider it to be zero because it is eliminated by filter.

Finally, we assume all the error sources are independent, thus the total error is estimated by:

$$e_{total} \approx e_{ref} + e_{weather} + e_{random} + e_{sys} \tag{5.10}$$

5.4 Evaluation and Discussion

We conducted a series of experiments to evaluate our method. Experiments consisted of the following three scenarios: outdoor, mountain and indoor. Since we cannot acquire authentic ground truth data, we compare our obtained elevation with DEM. Note that we only do this when the GPS accuracy is better than 6 meters. In our experimental area, Geospatial Information Authority (GSI) of Japan provides a highly accurate DEM (5m meshes). Nevertheless, this ground truth data still slightly different from actual elevation values. Also, we only evaluate the relative height for indoor case.

For base station reference points, we take advantage of meteorological stations of Japan Meteorological Agency which broadcasts weather data ev-

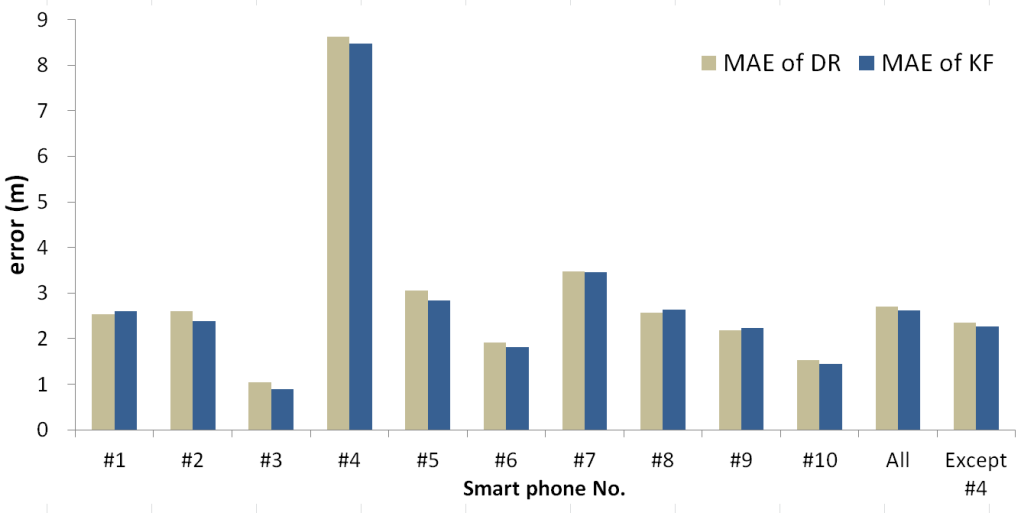


Figure 5.9: Elevation MAE on different smartphones by DR and KF

ery hour and provides history data off-line. Note that, DEM elevation refers to ground level while a smart-phone usually is held by a person. Therefore, in our system we add a constant (supposed 1.25 meter) to DEM elevation, to compensate for this.

We use different brands of smartphones (Samsung S3, S4 and LG Nexus 4) for experiments. In most experiments, air pressure from barometer is sampled in 20 Hz, and elevation is estimated every second. The error (difference between DEM elevation and estimated elevation by our method) is measured by MAE (mean absolute error), RMSE (root mean squared error) and MRE (mean relative error, which is divided by the elevation).

5.4.1 Case 1: Outdoor walking

In this case, 5 experiment participants with 10 smartphones walked in Setagaya district, Tokyo. Eeach participant walked about 2 hours on different routes (covered about 60 square km). The average elevation in this area is about 40 meters.

Figure 5.9 shows the MAE on different smartphones (different walking

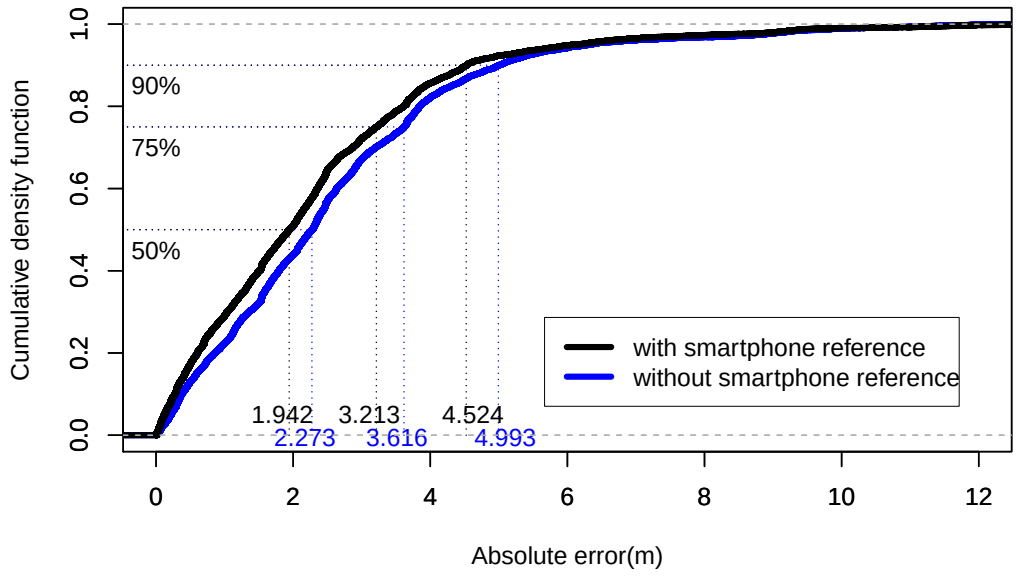


Figure 5.10: the CDF of elevation AE in Outdoor walking

routes). **KF** is the elevation obtained by applying Kalman filter while **DR** is the elevation obtained without Kalman filter (other processes are same). The label 'All' in the figure means the average of all smartphones, and the label 'Except #4' is the average of all smartphones except for smartphone #4. The MAE of most smartphone is very small (about 3 meters) but smartphone ♯4 is very large (about 9 meters). In fact, we found in the calibration test the variance of the barometer of smartphone #4 is very large, which is the reason why its result is abnormal. To avoid the error propagation of such malfunctioning barometer, the reference points server doesn't accept it as reference point. In addition, the result shows that **the elevation obtained by Kalman filter is generally better than without Kalman filter**. RMSE and MRE are not shown here, but they are less than 3.6 meters and 5% respectively.

Figure 5.10 shows the cumulative density of absolute error (AE) in all smart phones. This reveals that AE is less than 3.213 meters in 75% of the cases and less than 4.524 meters in 90%. Besides, we want to explore

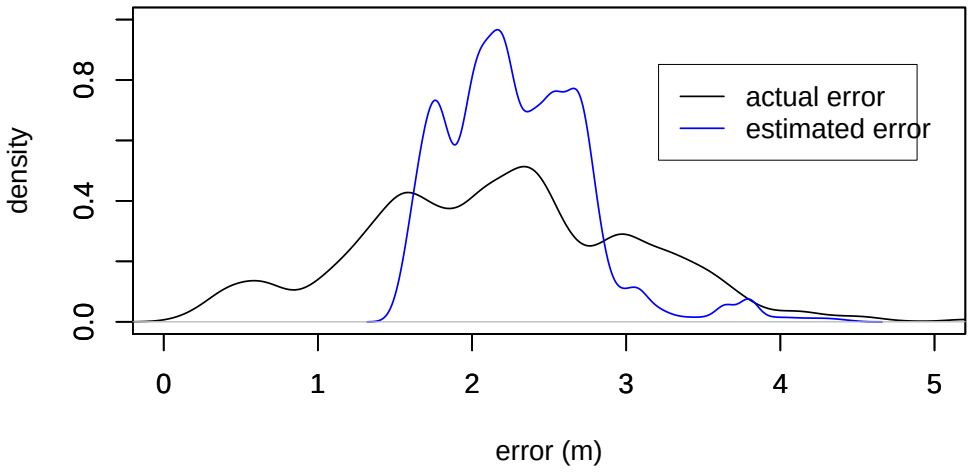


Figure 5.11: Density distribution of actual error and estimated error

accuracy of our method if we don't use any smartphone reference point. Evaluation results show that AE without smartphone reference is less than 3.616 meters in 75% and less than 4.993 meters in 90%, **meaning integration of smartphone reference improved accuracy by about 13%**. However, its worth noting that the contribution to accuracy by smartphone reference heavily depends on the setting of parameter *credit.sm*.

In this experiment, we also evaluated the quality of error estimation. We estimated the error on-line prior to the experiments by equation 5.10, and then calculated the actual error off-line. We compare what's the difference between estimated error and actual error. Figure 5.11 shows the density distribution of the actual and estimated errors. We find the estimated error shows similar trend as the actual error though there are some differences. In fact the mean difference between the two errors is about 0.651 meter and correlation is about 0.5.

Table 5.2: Elevation measurement errors in different mountains

Mountain	Peak Elevation (m)	MAE of DR (m)	MAE of KF (m)	RMSE of KF (m)	MRE of KF (%)
Mt#1	1859	4.63	4.36	5.44	0.3
Mt#1*	1859	4.74	4.46	5.54	0.3
Mt#2	599	3.72	3.74	5.14	1.1
Mt#2*	599	4.69	4.75	6.29	1.3
Mt#3	854	6.07	5.98	10.83	0.9

*: Does not integrate DEM reference

5.4.2 Case 2: Mountain climbing

In this scenario, experiment participants climbed 3 mountains (says $Mt\#1$, $Mt\#2$, $Mt\#3$ for convenience) on different days, the peak elevations are about 1900 m, 600 m and 850 m respectively. For $Mt\#1$ and $Mt\#2$, we did not have any smartphone reference. Results in Table 5.2 show that MAE on each of the mountains is very small (less than 6 meters), although this is worse than case 1 which was conducted in a low elevation area. The MRE is less than 1%, which is better than case 1. We still found that elevation by **KF** is more accurate than by **DR**. In addition, we compared the difference if DEM reference is integrated or not. In the case of $Mt\#1$ and $Mt\#2$ (table 5.2), **we found integration of DEM reference has advantage to a certain degree**. Similar to smartphone reference, the degree of accuracy improvement depends on setting of the parameter *credit.dem* and *cde* function. Note that, although the error of $Mt\#3$ (in which smartphone reference is used) is worse than other two mountains, it doesn't indicate that integrating smartphone reference works negatively, because the accuracy depends on the interaction of multiple factors and these 3 experiments are conducted in different environments.

Figure 5.12 shows DEM elevation, estimated elevation by our system and air pressure during mountain climbing ($Mt\#1$). Since DEM elevation (red line) and the estimated elevation (black line) completely overlapp, figure

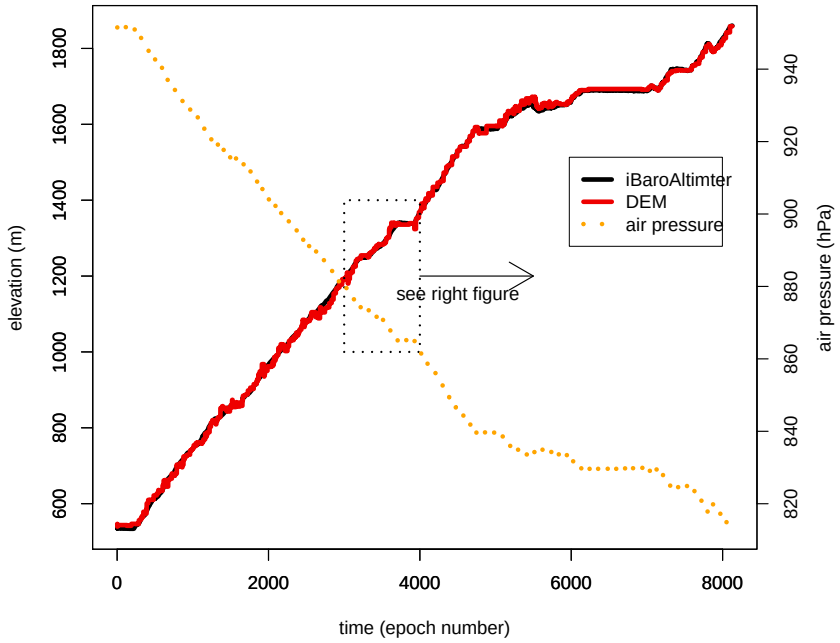


Figure 5.12: Elevation of DEM and our method in mountain climbing

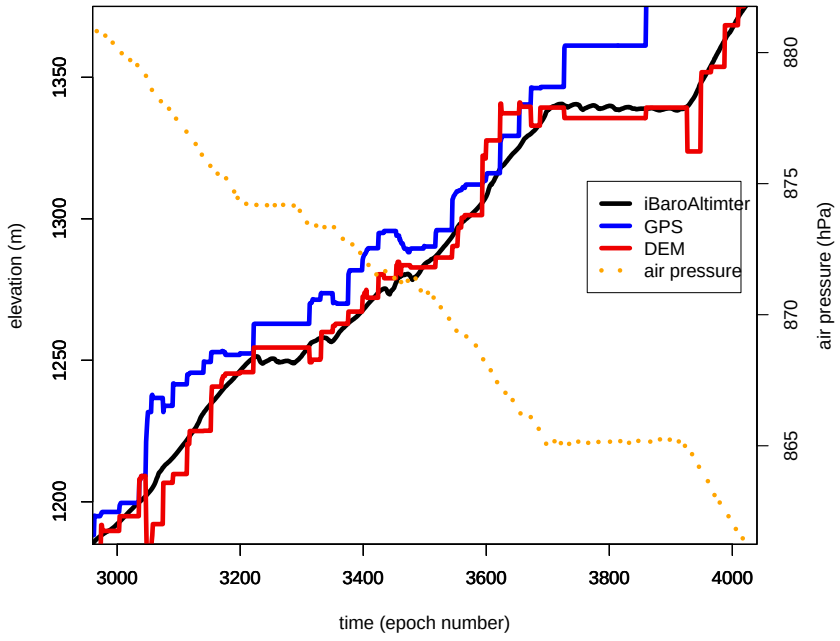


Figure 5.13: A part of elevation in mountain climbing

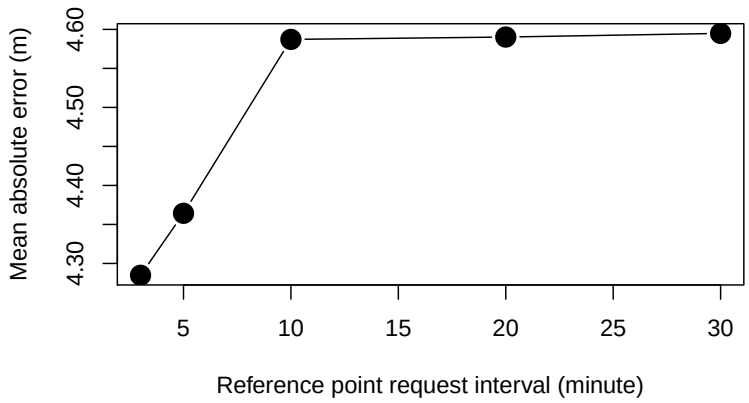


Figure 5.14: Error varies with interval of reference point request

5.13 which zooms into a section of the full trajectory (dotted box in figure 5.12) provides minute detail for easy understanding. Even though we don't know the real elevation profile, **we still can claim our obtained elevation profile is more reliable than GPS and DEM because it clearly looks more natural in mountain climbing**.

We also studied the effect of reference point request interval. **As shown in figure 5.14, MAE (in $Mt\#1$) increases as request interval becomes wide**. Incidentally, in our experiments request interval is set as 5 minutes.

5.4.3 Case 3: Inside buildings

In this case, we conducted experiments in 3 different buildings (says $Bld\#1$, $Bld\#2$, $Bld\#3$ for convenience), which have 8 floors, 15 floors, 45 floors respectively. Participants took stairs to go up and down in buildings $Bld\#1$, $Bld\#2$ while in building $Bld\#3$ they only used elevator. Inside buildings, there is no good GPS signals, consequently no $DEM\ reference$ is involved. We only evaluate the relative height of every floor by comparing with buildings' floor map. Figure 5.15 shows the height (relative to the basement floor) measured by our method in building $Bld\#1$. **From which, we find that**

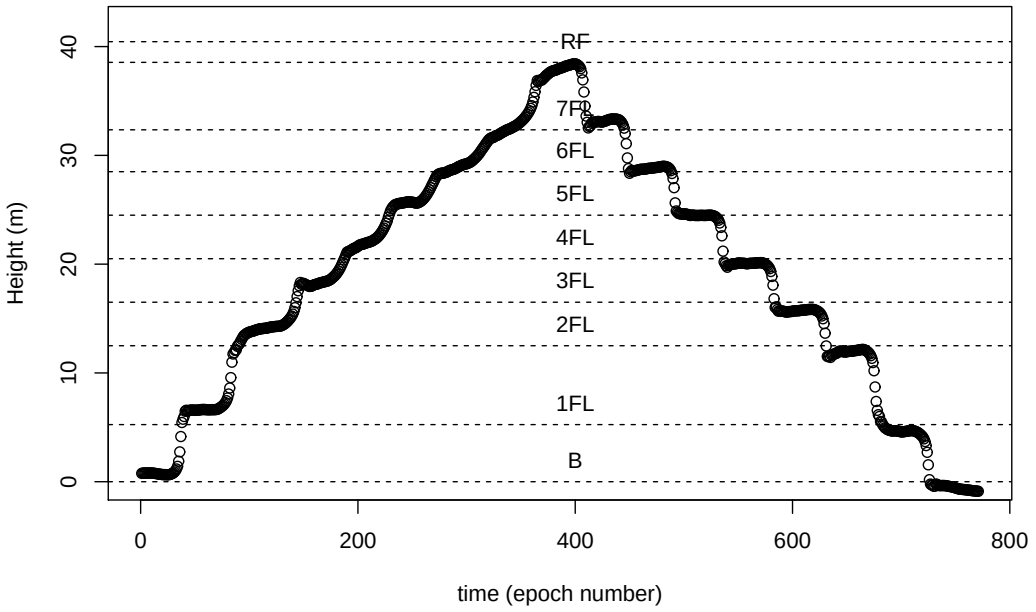


Figure 5.15: Indoor height measurement

the estimated height is equal to the floor map when going downstairs (right part of the figure), while it doesn't match the floor map perfectly when going upstairs(left part). Although we did not identify the cause of this difference, it is considered that going upstairs involves more irregular movement than going downstairs.

Figure 5.16 shows the MAE on different smartphones in different buildings ($Bld\#1, Bld\#2$). Here MAE is the error between our obtained height (relative elevation) and the height from floor map on every floor. We found the error is slightly different among smart-phones (different barometer). **The average error is about 0.86 meter**. For building $Bld\#1$, we did experiments on different days ($Bld\#2$ is on another day). We found the average accuracy on day #1 is marginally worse than on day #2. The probable reason is that the weather on day #1 (windy) is worse than on day #2 (sunny). Further, we found in this case, method **KF** doesn't perform better than **DR**, indicating that it is necessary to tune the process model of Kalman filter for

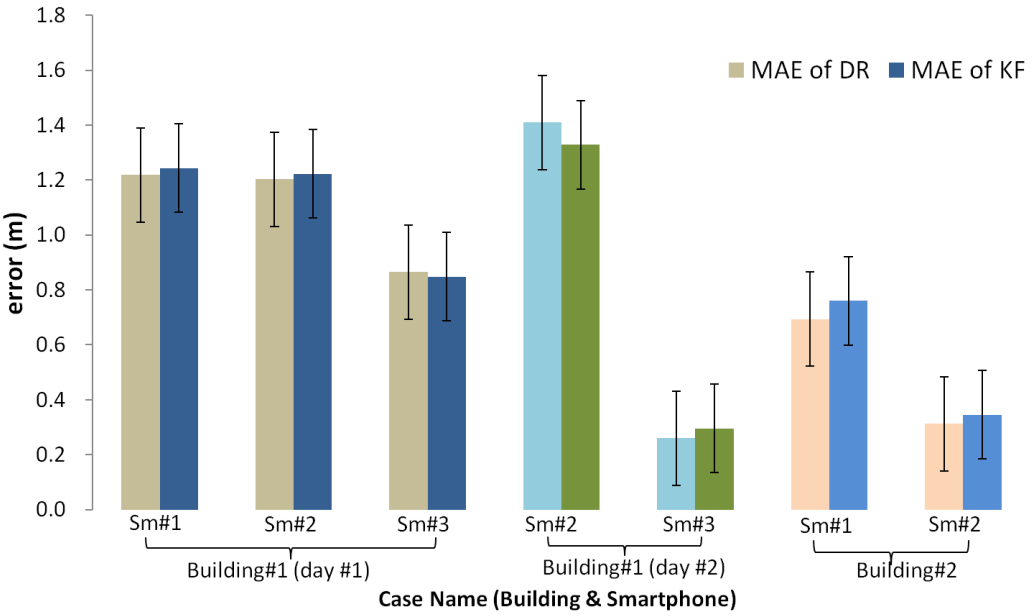


Figure 5.16: MAE of floor height on different smartphones in different buildings

complicated movements.

For building *Bld*#3 (45 floors), participants took elevator from 1st floor to the observatory (45th floor) and return for several times. Figure 5.17 shows an example of elevation measurement in this building. According to its public description, the height from 1st floor to 45th floor is 202 meters (taking 55 seconds by high speed elevator). We found that the relative height (from our estimated elevation) is roughly equal to the total height. Figure 5.18 shows the results on different smartphones (participants) in different rounds. The average error (labels 'All' in the figure) is about 2.5 meters, and the relative error is about 1.3%.

5.5 Related Work

Fusing barometer and GPS to improve the navigation system of aircrafts and UAV (Unmanned Aerial Vehicle) has been studied in various earlier works

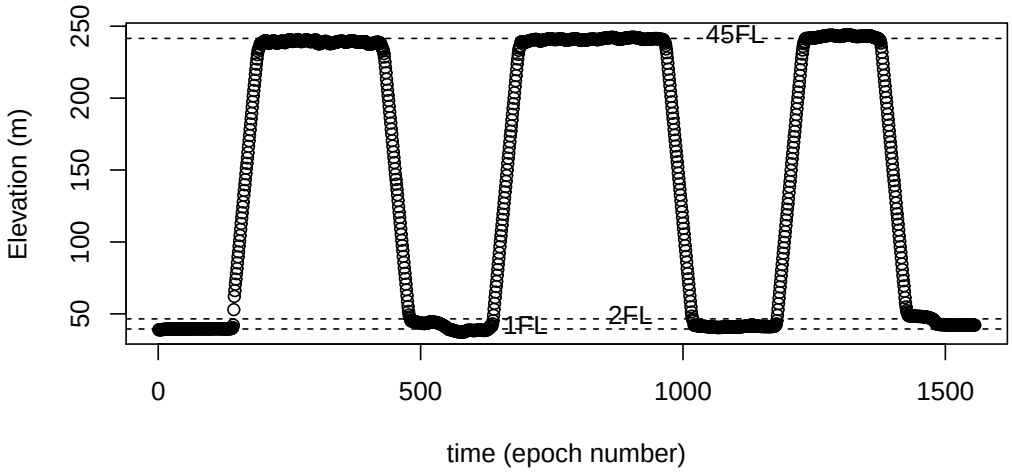


Figure 5.17: Elevation measurement in tall building (taking elevator)

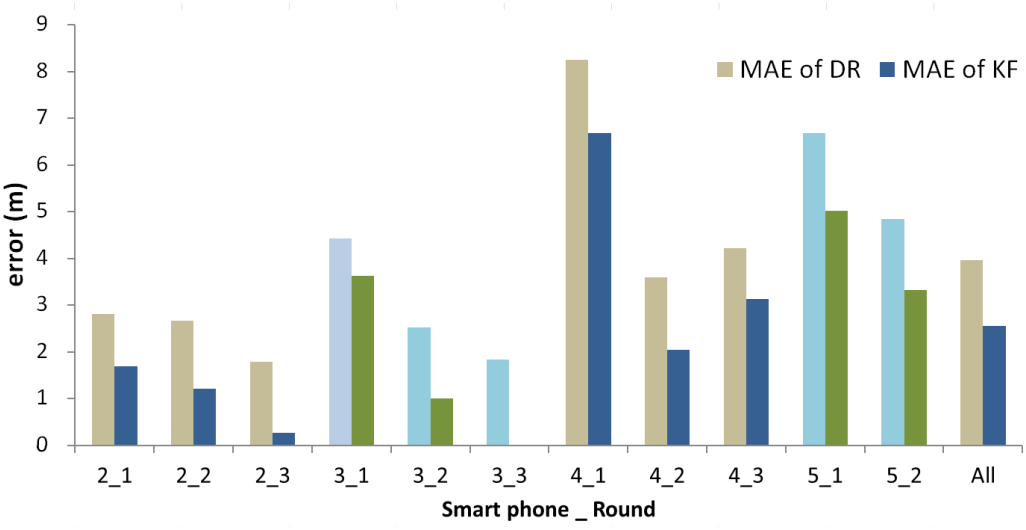


Figure 5.18: MAE of building height on different smartphones

[KS03, WLW12, GM95, ZS09]. Usually airports broadcast reference information, as such barometer based elevation measurement is accurate nearby airport. However, these technologies cannot be directly applied on smartphones in urban areas because reference point may be not within acceptable range and barometer of smartphones performs poor.

Numerous recent research works [PCL+11, HHD+12] have focused on indoor localization where GPS signals are not available, and 3D navigation. Li, Fan et al. [LZD+12] proposed a reliable indoor localization method using phone inertial sensors,while they only focused on the movement on the same floor, while it is easy to localize floors now by our method. Even before the advent of smartphone, some research works [AKWT12, CCE+12] had attempted to integrate barometer with mobile platform (e.g. laptop, sensor board) for floor localization. However, they conducted very simple experiments and only worked on relative height, which is less challenging because the error of relative height mainly depends on the quality of barometers. Recent research works [BPF+12, LHG13, KWS12] have attempted to utilize the embedded barometer sensors in smartphones for measuring height. Li et al.[LHG13] used airport control towers as reference points, and reported the accuracy is reliable only nearby airport, while our method provided reliable result in broader area. Keller et al.[KWS12] did good experiments which revealed that the barometers of smartphones are unstable and less accurate compared to high-performing barometers. It is also reported in [KWS12] that it is necessary to calibrate barometer sensor with respect to temperature, which we have not considered in our method yet. Jan et al.[JGEWE08] studied the empirical confidence bound for barometric altimeter errors and showed that incorporating this bound improves the performance of GPS-based approach and landing systems.

There are works[ZPZO13, ASD+09] on positioning based on GEO satellites. Due to poor geometry of GEOs, the availability and accuracy of positioning degrade rapidly, applying barometer can solve this problem. Ai et

al.[ASD+09] also proposed to create virtual reference points by interpolation from existing meteorological stations.

In addition, Srinivasan et al.[SDM10] proposed a method to interpolate atmospheric data by spatio-temporal kriging, which is similar to our spatio-temporal weighting method. However, the difference is that our method considers how air pressure changes affected the error in elevation while that method models air pressure by linear kriging. There are other works [PBGS00, LR04, JHP05] on spatio-temporal interpolation, however they are either too general or specific to some fields (e.g. soil water storage). Therefore, we cannot borrow their models directly.

5.6 Chapter Conclusion

We presented *iBaro-altimeter*, a system for accurate elevation measurement using barometers on smartphones. To the best of our knowledge, this is the first time to measure elevation (absolute height) with high accuracy using smart-phone. A series of experiments conducted in both indoor and outdoor environments with varying geographic characteristics using different models of smartphones, reveal that the errors are less than 3 meters in outdoor walking, 6 meters in mountain climbing, 0.9 meter in indoor floor localization. These errors are acceptable for most practical applications. For instance, in case of indoor floor localization, even with our recorded error of 0.9 m it is still possible to reliably detect floors.

However, to improve the accuracy of our method, we need to study the effect of key factors related to reference points such as topology, distance between reference points and target, and the number of smartphone reference points. In addition, we can optimize the parameters (i.e. dependent variables of weight w, such as N_s, $credit.sm$ and so on) to improve accuracy even further.

Chapter 6

Sensor Data Analysis

To explore collective intelligence is the main goal in crowd sensing. By aggregating sensor data from a group of participants, we can learn about human activities and urban environment. Next, three crowd sensing applications are independently explained.

6.1 Estimating Nighttime Activity

6.1.1 Problem Statement

Nocturnal lighting is a fundamental method for enabling human activity [EBD+99]. Outdoor lighting is used extensively worldwide in commercial, residential, industrial, public facilities, and roadways. There are numerous studies to discover knowledge from nighttime imagery, e.g. to estimate population density, global economic difference or electric power consumption [SREB01, LD12, LTZG08]. The main data source used in these works is from Defense Meteorological Satellite Program's Operational Linescan System (DMSP OLS), namely satellite image, and the main technique is remote sensing and image processing. Although they can analyze world wide data, the quality relies on the resolution of imagery, and satellite images are not easily acquired and it is impossible to process in real time mode.

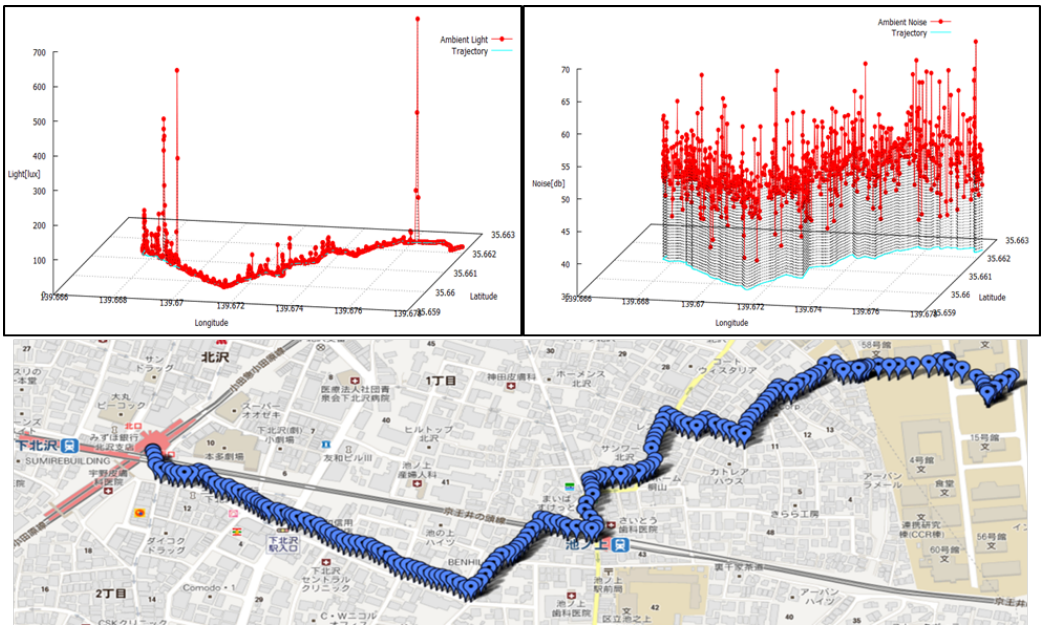


Figure 6.1: An example of collected data

Besides, ambient noise around a commercial area is mainly emitted from people conversation, passenger cars and commercial activities (e.g. hawking). Although it contains other sound sources, it still highly correlates with human activities, especially around densely populated area, e.g. in train station, shopping area, and public place of entertainment.

In sum, ambient light and noise at night are the energy emitted from the area through human activities. Consequently, we claim the ambient light and noise can indicate how active an area's night-life is.

6.1.2 Proposed Solution

By making use of the microphone and the light sensor built-in smartphone, we developed a tool based on Android OS. Participants take smartphone installing this tool to collect ambient noise and ambient light along their routes during daily-life. Through crowd sensing, we can collect fine-grained data at a large scale. Figure 6.1 shows an example of collected data, in which

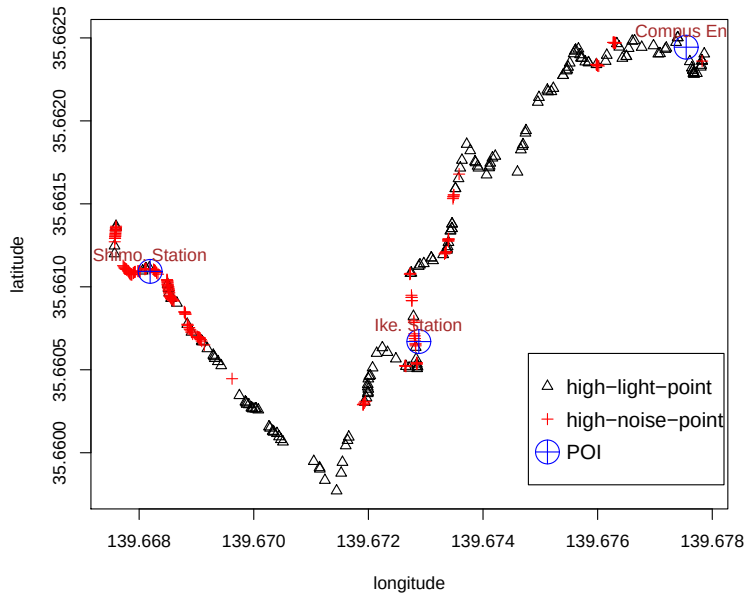


Figure 6.2: An example of spatial correlation between high-light points and high-noise points

the upper-left is the illumination intensity (*lux*), the upper-right is the noise level (*decibel*) and the bottom part is the sensing trajectory on the map.

Due to the difference of sensors among different models of smartphone, the raw data has to be calibrated to remove the error. In addition, the value of light sensor is subject to the orientation of the smartphone with respect to the light source. Since we also record the orientation of smartphone in real-time, it is possible to adjust the value according the relative orientation while we do not adjust in this work just because the inaccuracy caused by orientation is allowable when we roughly estimate how busy the area is.

Another major concern is whether or not ambient noise and ambient light depends on each other. Since the data is inherent to spatial position, it is not adequate to only calculate the correlation value. Therefore, we employ Ripley's *K function* [Rip05] to find the distribution relationship. *K function* is based on the areal density of points. However, we collect the data along

trajectory, which means we can only measure the linear density. Thus, we modify *K function* as follows:

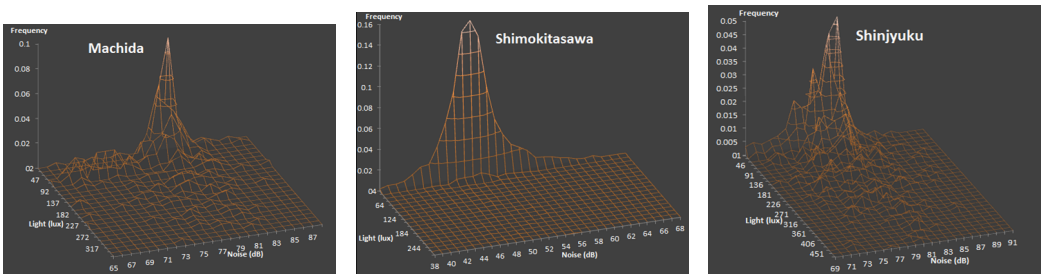
$$\begin{aligned} L_{ij}(h) &= \frac{K_{ij}(h)}{2} - h \\ \lambda_j K_{ij}(h) &= E(|\{P_j : dist(P_i, P_j) < h\}|) \end{aligned} \tag{6.1}$$

where $P_i \in$ Noise Points, $P_j \in$ Light Points, λ is linear density, h is the distance (m). Note that, in traditional way *K function* is defined as an area function (i.e. πh^2), while here we define it as a length function (i.e. $2h$).

By using the data collected in our project, we calculated the value of $L_{ij}(h)$ is close to zero, which indicates that ambient noise and ambient light are independently distributed random variables. In short, the noisy place does not necessarily mean bright, vice versa. The figure 6.2 shows high-noise place (top 10 percent of the entire points) and high-light place along a trajectory, which can substantiate that high-noise points (see + in the figure) don't always collocate with hight-light points (see △ in the figure). Consequently, we can employ both of them to estimate how busy a city is. Incidentally, ambient noise approximately follows normal distribution while ambient light approximately follows exponential distribution if without consideration of spatial feature.

6.1.3 Experimental Result and Discussion

After collecting data via our trajectory sensing project (see chapter 2), we made the following data analysis. Here, we chose 3 sampling files for discussion. Figure 6.3 shows the light-noise joint distribution of these 3 areas, where (a) is around Shinjuku Station, Tokyo, (b) is around Machida Station, and (c) is from Komaba campus of UT to Shimokitazawa Station. Just checking the shape of the distributions, we can find area (a) and area (b) are more active than area (c). To precisely measure the similarity between two areas, we tried two methods. The first method is to classify the light-noise



(a) (b) (c)

Figure 6.3: Light noise joint distributions around (a) Machida station, (b) Shimokitazawa station, (c) Shinjyuku station.

of each area into k clusters by K-means. Then calculate the cosine similarity of clusters between each two areas by the definition introduced in previous work (see section 4.4.3 of chapter 4). According to the result, we found that area (a) is more similar to area (c) than to area(b), which is consistent with our common sense about these 3 areas. The second method is to calculate *Jensen-Shannon Divergence* of light-noise distributions between each two areas. The result shows the same conclusion as did in the first method. However, to validate the accuracy of the result, we need to compare with people flow data or questionnaire in future.

6.2 Noise Maps

6.2.1 Introduction

Environmental noise (ambient noise) is a crucial factor to our health and working efficiency. Noise pollution draws attention from many researchers [Kan10, TLC09, MSNS09]. They try to find noise pollution at different area in different time period. In addition, other researchers make efforts on exploring ambient noise to sense events. For example, Rijurekha et al. [SSR11] did work on traffic condition monitoring by using ambient noise measured by self-designed sound meter. Rossi et al. [RFA+13] presented a sensing system to recognize user context by analyzing ambient sound sampled from smartphone's microphone.

In our work, we also leverage the ubiquitous smartphone to build a noise map for a community. Similar to some existing work [Kan10, MSNS09], we applied crowd sensing approach, which is less costly and can collect data at a large scale with fine granularity. We conducted a large scale noise sensing experiments in Setagaya-Ku, the detail is stated in chapter 2. In addition, in this work we have an additional purpose which is to specially acquire data around about 2500 measurement places. These measurement places consist of 718 factories and 1714 residential houses in that area. The ambient noise data (with other sensor data) around these specific places is also used for another application which is out of the scope of my thesis.

6.2.2 Building Noise Map

As stated before, there are about 2500 measurement points through which participants will walk. In participatory sensing or crowd sensing, it is important to motivate participants to join sensing activities. Thus, with the consideration of fairness we need to equally assign all measurement points to each participant. In addition, total walking distance should be minimized so

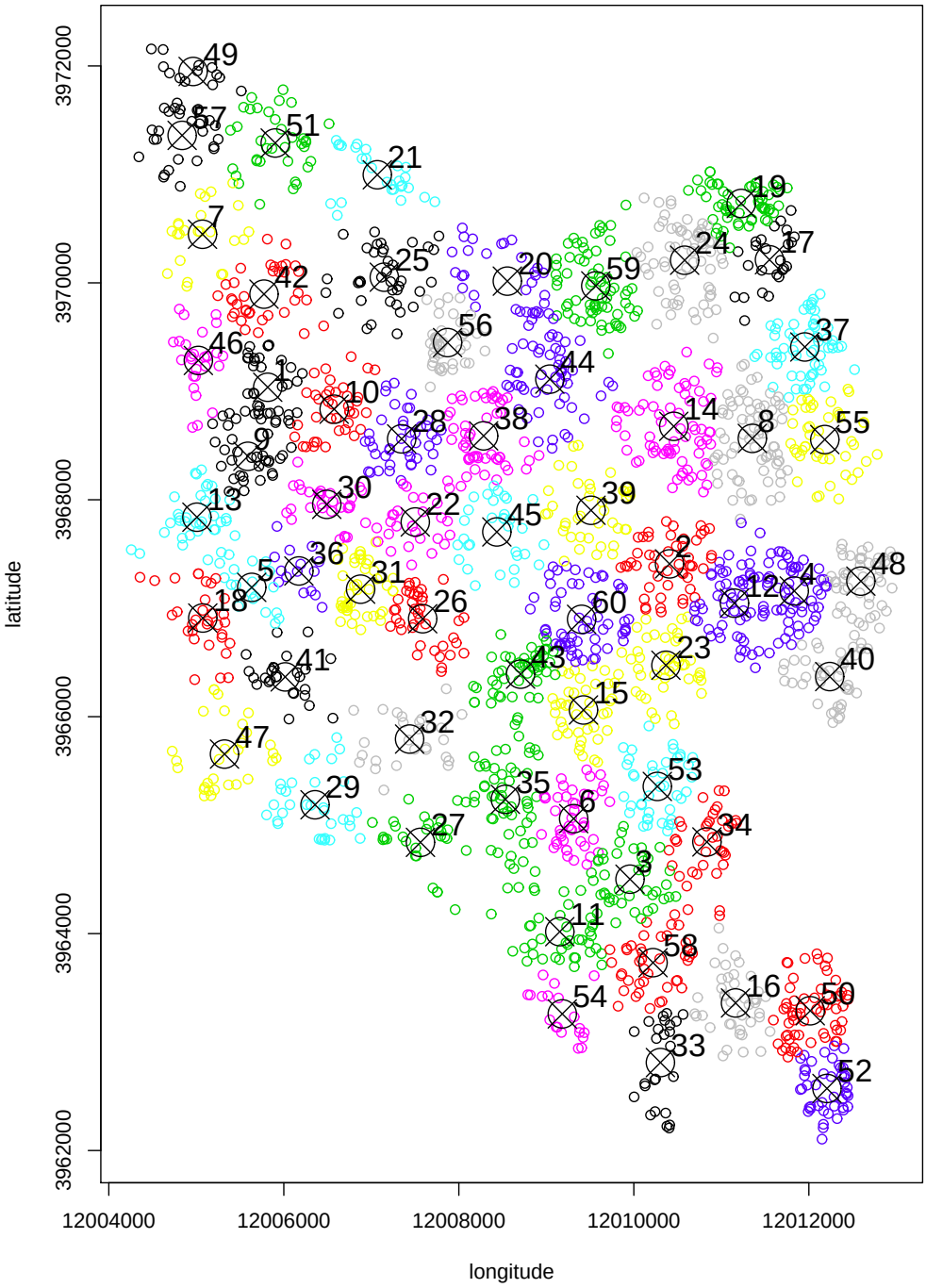


Figure 6.4: Clusters for participants on measurement points

that we can acquire all data with minimum workload. Therefore, we applied $K - means$ to divide all measurement points to clusters, then assign each cluster to each participant per day. Figure 6.4 shows the clusters for assignments, in which 60 clusters are created for 40 participants (some participants attended multiple days). Although the number of points in each cluster is not consistent, the walking distance of each cluster is nearly equal so that it guarantees the fairness for participants.

A microphone converts pressure fluctuations into an electrical signal (called sound pressure P_v) that can be used to compute the loudness of the noise source. Sound pressure P_v is a logarithmic measure of the effective sound pressure of a sound relative to a reference value. It is measured in decibels (dB) above a standard reference level. The standard reference sound pressure (P_{ref}) in air is 20 Pa. Accordingly, the sound level L_p is computed as:

$$L_p = 10\log(\frac{P_v^2}{P_{ref}^2}) = 10\log(P_v^2) + 93.9794\ (dB) \tag{6.2}$$

Because the human ear doesn't respond to all of these frequencies equally well, weightings are applied to the sound levels measurements (in the frequency domain) to take into account this selective behavior of the human hearing system. We also applied A-weighting (regulated by an international standard IEC 61672) to adjust sound levels by frequency in effort to account for the relative loudness perceived by the human ear.

In addition, smartphone's microphone has systematic error when it is used to measure noise level. Therefore, before using it, we need to calibrate it. Our calibration is done at an anechoic chamber where is echo-free (see figure 6.5(a)). The calibration test follows the procedure: (1) set a predefined relative sound intensity on the speaker, (2) play a short time Brownian noise (also known as red noise) on the speaker, (3) record the sound level (dB) on the professional sound meter (see figure 6.5(b)), (4) record the sound level on all calibrating smartphone (see figure 6.5(c)), (5) repeat (1)-(4) at different

(a) anechoic chamber (b) sound meter (c) smarthphones

Figure 6.5: Calibration test for smartphone's microphone at anechoic chamber.

relative sound intensity. Note that, the reason why we use Brownian noise to calibrate is that Brownian noise is closer to actual ambient sound than white noise and others.

6.2.3 Experimental Result and Discussion

Following the calibration procedure aforementioned, we have done the calibration test. As shown in figure 6.6, the result shows that all 4 different smartphone models have their particular systematic bias comparing to the standard sound meter (labeled as 'Input' in the figure). The average of systematic bias is about 3.1 dB, and it increases as sound level goes up. This result is used to compensate the raw value getting from smartphone when building noise map.

As the experiment for this work is elaborated in chapter 2, here we will skip the detail of experiments but report the result. Participants record ambient noise with position (from GPS) along their traces, thus the original data is linked to trajectories (see figure 6.7(a)). However our purpose is to map ambient noise on specific measure points, as a result we create a noise map by visualizing the noise level at every measure point (see figure 6.7(b)). The noise level at every measure point is computed by taking average value of N nearest sample points from original trajectory data. When we zoom in

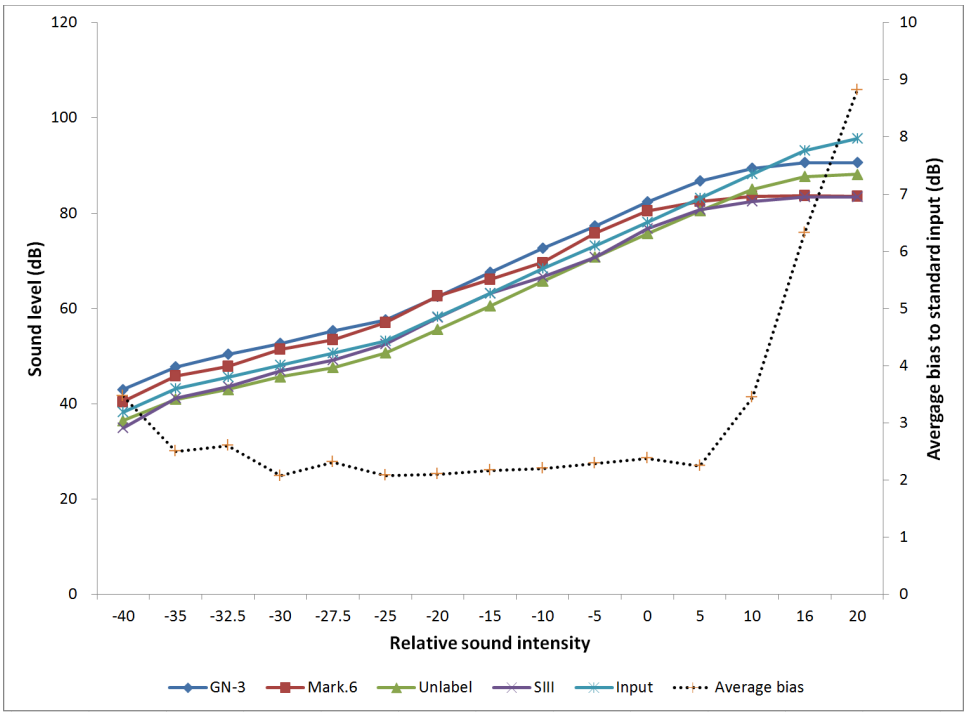
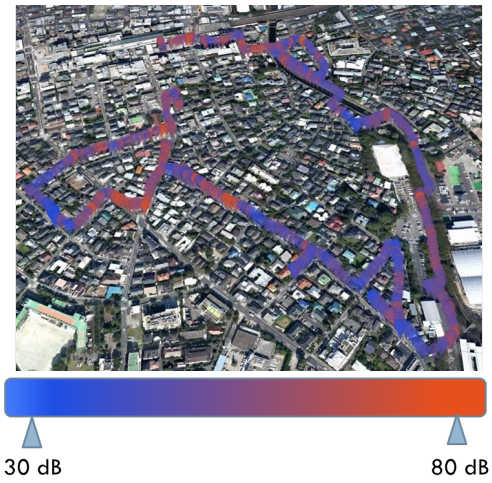
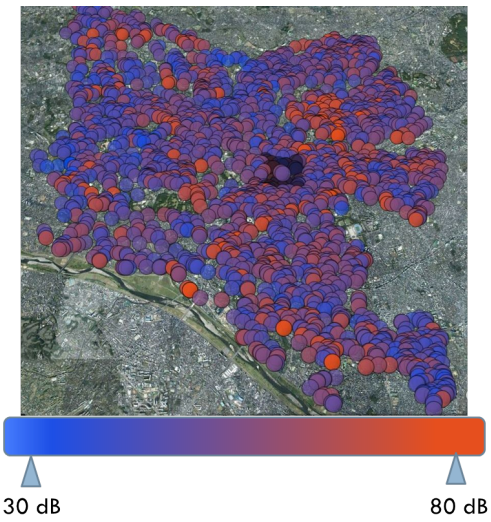


Figure 6.6: Calibration results for different models of smartphone

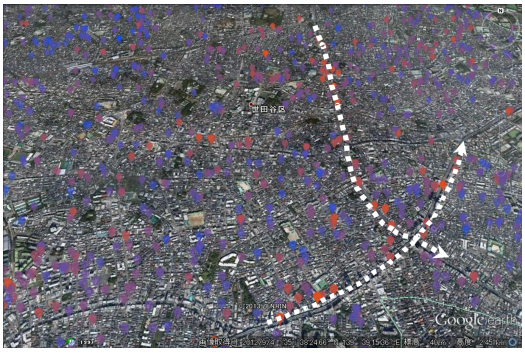
the full map (figure 6.7(b)), we can find the detail of noise level. Figure 6.7(c) shows that noise level along expressways (two white dot lines in the figure are national expressway $Road-246$ and Tokyo circular route $Road-318$ respectively) is very high.



(a) trajecoty with noise level



(b) full noise map



(c) zoom in express ways

Figure 6.7: Noise map for Setagaya-Ku.

6.3 Weather Sensing

6.3.1 Problem Statement

Nowadays smartphones are explored to sense the physical world via mobile crowd sensing [GYL11] or participatory sensing [LMLe10, CELea06]. On the other hand, wireless sensor networks (WSN) for environment monitoring has been well studied for a long time [ASSC02]. There are an array of disadvantages and advantage in these two sensing means.

Mobile sensing, especially by crowd sensing approach, can recruit numerous participants to sense in a large area with fine granularity. It is opportunistic sensing, meaning less costly. However, sensors embedded in smartphones don't perform as good as these dedicated sensors used in wireless sensor network or special sensor networks. For example, air pressure sensors of smartphones have large systematic error and are unstable by comparisons to ones of meteorological stations (for weather forecast services). In contrast, sensors in those dedicated sensor networks possess good performance while they are expensive, static and sparse.

Therefore, it is a promising way to integrate both these mobile sensors and fixed sensors for sensing. However, it is challenging to cope with the dynamic and heterogeneous features of this combined network. In this work, we try to sense and predict the sensor values in such a hybrid sensing systems.

In most countries, meteorological stations are set up to observe weather conditions (e.g. temperature, humidity, air pressure, rainfall, solar radiation). However these stations are sparsely located, thus it is difficult to provide services in micro level (called micro-climate services). In Japan, for example, there are 846 stations run by Japan Meteorological Agency, whose spatial intervals are about 20 kilometres. In addition, these stations periodically broadcast weather information at a specific time interval (every hour). To acquire finer granular weather information, we introduce smartphones which are equipped with weather sensors to measure air pressure, temperature and

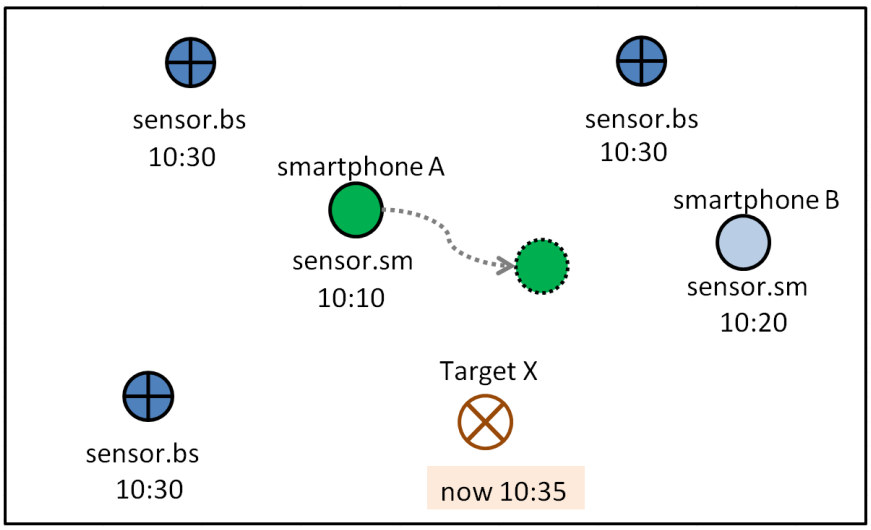


Figure 6.8: An illustration of heterogeneous network where mobile sensors and fixed sensors coexist

humidity. Through an approach of crowd sensing, we can collect the sensor data with fine granularity in terms of both time and space.

As shown in figure 6.8, static sensors from meteorological stations (*sensor.bs*) broadcast weather information every fixed time interval (usually 1 hour), while mobile sensors from smartphones (*sensor.sm*) dynamically obtain data (dynamic location and time). In addition, we also have to consider the accuracy of these two types of sensors. Note that, for the sake of saving energy, smartphones send sensor data to back-end server at a specific time interval (usually 5 minutes). Now, our goal is to predict the value at any location (e.g. $TargetX$ in figure 6.8) at any time.

6.3.2 Proposed Method: Integrating Mobile Sensors with Static Sensors

Our basic idea is to predict the value by spatio-temporal interpolation from all mobile sensors and fixed sensors. There are related work in the literature while they only deal with problems in a homogeneous sensor network

[MLR12, VAA04]. When a sensor node is closer to the target location, it is reasonable to weight it more for interpolation. In a similar way, the newer sensor data will be weighted more too. Thus, we can take into account these three factors: horizontal distance, vertical distance and time gap, to assign a weight on each sensor. Each factor independently leads to error in a certain form (let's say $f(hs), g(vs), h(t)$ respectively). We assume their form follows a exponential function (Gaussian distribution) which can be verified by approximating the error distribution with regression method. Therefore, to integrate all sensor nodes, each sensor node is given a weight w based on spatio-temporal distance as follows:

$$\begin{gathered} w = f(hs) * g(vs) * h(t) \\ assume\ f(hs),\ g(vs)\ and\ h(t)\ follow\ N(0, \sigma_{hs}^2), \\ N(0, \sigma_{vs}^2),\ N(0, \sigma_t^2)\ respectively \\ w = (2\pi)^{-\frac{3}{2}} (\sigma_t \sigma_{hs} \sigma_{vs})^{-\frac{1}{2}} e^{-\frac{1}{2}[\frac{t^2}{\sigma_t^2} + \frac{hs^2}{\sigma_{hs}^2} + \frac{vs^2}{\sigma_{vs}^2}]} \end{gathered} \tag{6.3}$$

where t is the time gap from the sensor information time-stamp to current time, hs is the horizontal distance (calculated from latitude and longitude), vs is the vertical distance (elevation). In addition, since the accuracy of fixed sensors is better than smartphone sensors, the weight w will be multiplied by a constant according to the type of sensors. Note that, this interpolation method is also introduced in my previous work (in chapter 5) for elevation measurement from air pressure.

6.3.3 Experimental Result and Discussion

To validate our method, we conducted some preliminary experiments. Several participants carry the smartphones which install our developed sensing tool, and walk in Setagaya district, Tokyo. We also make use of meteorological stations of JMA to receive weather information. As shown in figure 6.9, we have two static stations (*sensor.bs*) and two smartphones (*sensor.sm*).

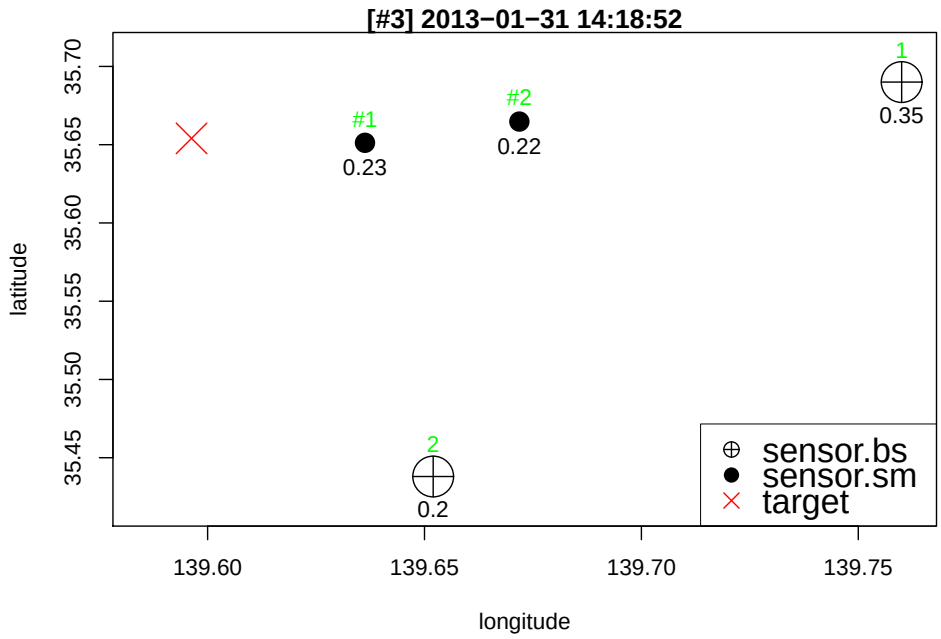


Figure 6.9: An example of dynamic network in our experiments

Another smartphone (*target*) is used as ground truth to evaluate the predicted value, meaning target location is mobile. In this experiment we only sensed **air pressure**, though our method will work in terms of temperature and humidity too.

Figure 6.10 shows the CDF of absolute error which is the difference between the measured air pressure from smartphone and predicted value from our method. The absolute error is 21 pascal in 75% of cases and 44 pascal in 95% of cases (see blue line in figure). Note that the measured air pressure is not authentic ground truth data since it is measured by smartphone. On the other hand, there is a large error (grey line in figure) when the smartphone used as ground truth is not calibrated. Note that it doesn't indicate our method is ineffective but substantiates the smartphone sensors need to be calibrated. In addition, figure 6.11 shows an example of measured air pressure and estimated air pressure, in which the mean absolute error is about 0.123 hPa. Incidentally, since our estimated air pressure (black line in the

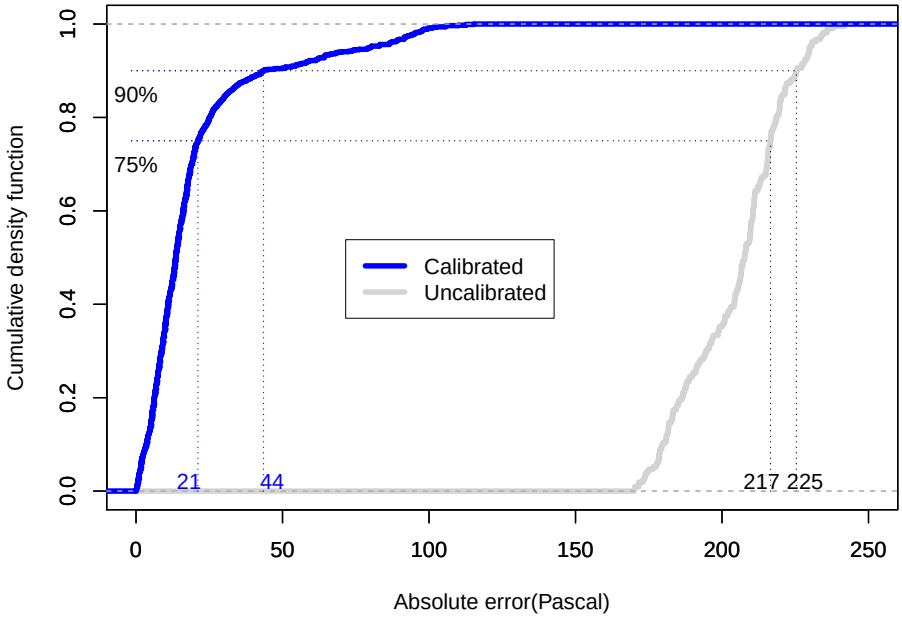


Figure 6.10: Absolute error between predicted air pressure and measured air pressure

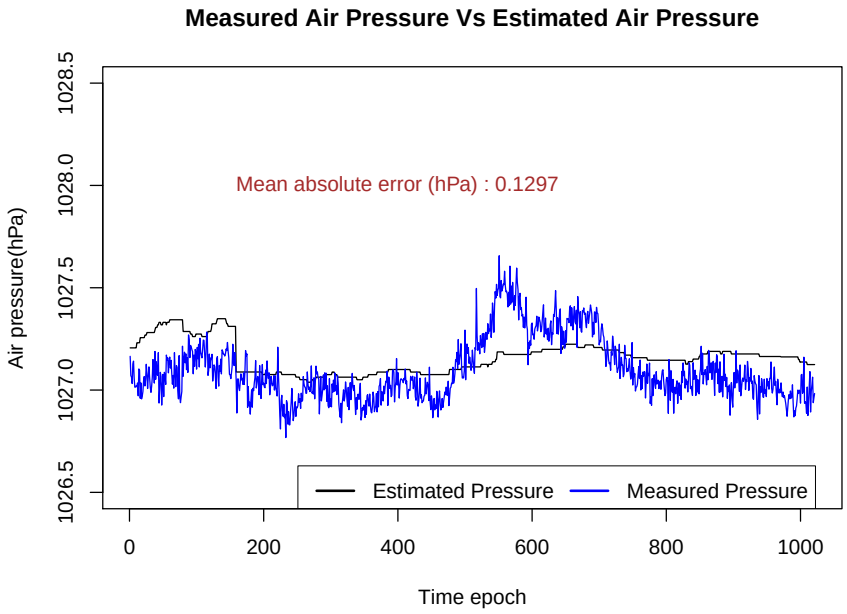


Figure 6.11: Example of measured air pressure and estimated air pressure

figure) is processed by Kalman filtering, it is smoothed. In contrast, the raw measured air pressure is jerky probably due to the instability of smartphone sensor.

We proposed a spatio-temporal weighting method to predict sensor values in a heterogeneous network where mobile sensors and fixed sensors coexist. Preliminary experiments prove that our method can reach a good result. However, to improve the accuracy even further, we need to optimize parameters, study the effect of the topology of sensor nodes. In addition, it is more meaningful to leverage this approach on predicting temperature and humidity in future.

Chapter 7

Conclusion and Future Work

Thanks to the development of sensor technology and the high penetration of smartphone, it has put forth a new sensing approach - crowd sensing. Crowd sensing with mobile devices brings about both opportunities and challenges. In this thesis, we studied how to leverage crowd sensing to analyze human activities and urban environment, including system architecture design, sensor data management, special sensor application, several data analysis cases.

As for the concerned 3 problems stated in chapter 1,

- How to utilize smartphone sensors and manage sensing activities?
  We developed a dedicated sensing tool on smartphone (see chapter 2), which fully leveraged the sensing power of smartphone and easily manage many participants for a specific or general sensing activity.
- How to handle huge volume of sensor data?
  We proposed a trajectory simplification method to speed up trajectory analysis (see chapter 3), and a sensor data management scheme to reduce data volume while keeping information content (see chapter 4).
- How to acquire highly accurate sensor data?
  In our sensor application (iBaro-altimeter) we challenged to obtain accurate elevation from poor sensors of the smartphone (see chapter 5). We also tried to analyze data for finding reliable knowledge even though

the raw sensor data is not stable and accurate (see chapter 6).

More specifically, we first designed our sensing platform for data collection and data analysis, which enable us to sense the physical world and human society at a large scale. A sensing tool on smartphone is developed to collect all kinds of sensor data, such as ambient noise, light, location, acceleration etc. Then we conducted a large scale sensing experiments by using this tool.

Trajectory obtained from GPS is the key factor to analyze human mobility and transport issues. Trajectory simplification can greatly improve the efficiency of data analysis (e.g., trajectory search, trajectory clustering). we proposed a novel scheme: divide/merge principle and selection strategy to reduce data for spatial trajectory. To measure displacement correctly, we newly introduced enclosed area metric which is proven more robust against GPS uncertainty. In trajectory similarity computation, this metric is also superior to conventional distance-based metric. Through comparing with other methods in a series of experiments over huge dataset, our method is proven effective and efficient. Furthermore, our method is a pure online procedure which can be readily installed at the mobile terminal to preprocess trajectory before sending it to back-end server.

We also solved another major issue: the sheer volume of data collected through crowd sensing can deeply hamper the performance of various applications. It is fairly challenging and significant to extract more useful data but reduce redundant data. We proposed a data model (REPSense) to represent data from the viewpoint of data diversity. We rethink data reduction problem by analogy to electoral system. In our method, data is selected by a fashion which is similar to electing congressional representatives. Although we don't theoretically prove our model is effective, we draw our conclusion based on a large amount of experiments with real data set. Through comparing with conventional methods and state-of-the-art methods, our method performs well in terms of data divergence especially Itakura-Saito distance. Our method also outperforms other methods in terms of clustering perfor-

mance. It is foreseeable that data reduction will linearly mitigate storing and transmitting problems as mentioned in the first section, though we did not evaluate it by experiments.

Furthermore, we developed a practical system (called iBaro-altimeter) to measure elevation by using smartphone's barometer. Accurate elevation is extremely useful in many scenarios, such as 3D navigation, floor localization, emergency rescue, etc. It is possible to use barometers on smartphones to estimate elevation in both indoor and outdoor scenarios. However, to reach an acceptable level of accuracy, a reference point which periodically broadcasts its air pressure and temperature is required. We proposed a method to increase the spatio-temporal density of reference points by exploiting neighboring smartphones as ad-hoc reference points. In addition, we also employed Kalman filter to stabilize the elevation profile due to the instability from smartphone sensors. A series of experiments conducted in both indoor and outdoor environments with varying geographic characteristics using different models of smartphones, reveal that the errors are less than 3 meters in outdoor walking, 6 meters in mountain climbing, 0.9 meter in indoor floor localization. These errors are acceptable for most practical applications.

Last but not least, we analyzed sensor data for specific purposes: (1) estimate how busy an area is by using ambient light and noise from smartphone sensors; (2) build noise map for a residential area to find noisy area; (3) sense and estimate micro-scale weather by integrating smarphone's weather sensors with meteorological stations.

There is a lot of work left for us since crowd sensing is still at its infancy. For architecture design, we still lack scalable and energy-efficient platform. In addition, privacy preservation and incentive mechanism are still an urgent problem to be solved, before crowd sensing fully exerts its power. In most current work, researchers still only utilized a small array of sensors, mainly GPS, camera, microphone, acceleration. By integrating all other sensor data, there are numerous potential applications. In future, we believe that new smart

wearable devices will become ubiquitous, such as, smart glass, smart watch and smart clothes. These mobile devices with strong sensing capability make people more easily involve into sensing, which will boost the development of crowd sensing. By that time, everyone can participate in sensing everything at every-time and everywhere.

Appendices

Appendix A

Publication List

A.1 Journals

1. Guangwen Liu, Masayuki Iwai and Kaoru Sezaki, "An online method for trajectory simplification under uncertainty of GPS", IPSJ Transactions on Databases, vol. 6-2, June, 2013.
2. Muhammad Asif Hossain Khan, Danushka Bollegala, Guangwen Liu, and Kaoru Sezaki, "Delineating Real-Time Events by Identifying Relevant Tweets with Popular Discussion Points", ASE Human Journal, Vol. 2, No. 3, pp. 136-150, 2013.

A.2 International Conferences

1. Guangwen Liu, Masayuki Iwai and Kaoru Sezaki, "A Method for Online Trajectory Simplification by Enclosed Area Metric", International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2012), Okinawa Japan, May 2012.
2. Guangwen Liu, Masayuki Iwai, Yoshito Tobe and Kaoru Sezaki, "REPSense: On-line Sensor Data Reduction for Mobile Sensing by Preserving Data Diversity", IEEE WiMob 2013, October, France, 2013.

3. Masayuki IWAI, Houhei Shigeta, Guangwen Liu, Shunsuke Aoki and Kaoru Sezaki, "A System for Large-scale Participatory Environmental Noise Sensing", The SICE Annual Conference 2013, September, Nagoya, 2013.

4. Muhammad Asif Hossain Khan, Danushka Bollegala, Guangwen Liu, and Kaoru Sezaki, "Multi-Tweet Summarization of Real-Time Events", ASE/IEEE International Conference on Social Computing, Washington D.C., USA., September, 2013.

A.3 Domestic Conferences

1. Guangwen Liu, Masayuki Iwai, Masaki Ito and Kaoru Sezaki, "Weather Sensing in Heterogeneous Network Where Mobile Sensors and Fixed Sensors Coexist", 2014 IEICE General Conference, vol. BS-1-5, Niigata, Japan, March, 2014.

2. Dunstan Matekenya, Guangwen Liu, Masaki Ito, Kaoru Sezaki, "Energy Efficient Sensing with Spatio-temporal Predictioni", 2014 IEICE General Conference, Niigata, Japan, March, 2014.

3. Guangwen Liu, Masayuki Iwai and Kaoru Sezaki, "Estimation of How Busy a City is by Mobile Sensing", 2013 IEICE General Conference, vol. B-19-36, Gifu, Japan, 19-22 March, 2013.

4. Guangwen Liu, Masayuki Iwai, Kaoru Sezaki, "A Fast and Novel Online Compression Method for GPS Trajectory," IEICE General Conference, A-17-11, March 2012.

5. Guangwen Liu, Masayuki Iwai, Kaoru Sezaki, "A Preliminary Study on Trajectory Sensing and Data Reduction in Mobile Terminal," IEICE Human Probes Workshop, February 2012.

6. 青木 俊介, 劉 広文, 岩井 将行, 瀬崎 薫，"ユーザ固有の雑音を考慮する参加型環境センシングのデータ校正手法", IPSJ UBI 38 回, 熊本, 2013

年 5 月.

7. 青木 俊介, 劉 広文, 清水 和人, 岩井 将行, 瀬崎 薫，”ユーザ参加型環境センシングにおける効率的なシステム運用モデルの構築とユーザ分析”, DICOMO2013, 2013 年.
8. 重田 航平, 青木 俊介, 劉 広文, 岩井 将行, 瀬崎 薫，”モバイル端末を用いたユーザ参加型環境センシングにおけるデータ欠損の検知および処理に関する考察”，DICOMO2013，2013 年 7 月.
9. Muhammad Asif Hossain Khan, Guangwen Liu, Masayuki Iwai and Kaoru Sezaki, ”What Does Chirping Tell Us? Summarizing People's Opinion on Ongoing Events Using Tweets”, Life Intelligence and Office Information Systems, IEICE Technical Report, Vol. 112, No. 466, pp. 107-112, Miyakojima, Japan, 7-8 March, 2013.
10. 岩井将行，柳原正，清水和人，澤上佳希，劉　広文，瀬崎　薫，”運転者の移動履歴及び移動意図記録システムの提案と実装”，情報処理学会全国大会，6D-6，2012 年 3 月.
11. (Poster) Guangwen Liu, Masayuki Iwai and Kaoru Sezaki, ”Trajectory Sensing by Human Probe”, 12th Academy of Human Informatics, Tokyo, 11 September, 2012.

Bibliography

[10N] Noisetube. http://noisetube.net/.

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[AKWT12] C Ascher, C Kessler, R Weis, and GF Trommer. Multi-floor map matching in indoor environments for mobile platforms. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–8. IEEE, 2012.

[ASD+09] GuoXiang Ai, PeiXuan Sheng, JinLin Du, YongGuang Zheng, XianDe Cai, HaiTao Wu, YongHui Hu, Yu Hua, and XiaoHui Li. Barometric altimetry system as virtual constellation applied in caps. *Science in China Series G: Physics, Mechanics and Astronomy*, 52(3):376–383, 2009.

[ASSC02] Ian F Akyildiz, Weilian Su, Yogesh S., and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[BEHea06] J. Burke, D. Estrin, M. Hansen, and et al. Participatory sensing. In *WSW Workshop at SenSys*, 2006.

[BPF+12] A Bahillo, J Prieto, H Fernandez, P Fernandez, RM Lorenzo, and EJ Abril. Fusing technologies for a continuous positioning

solution developed on a smartphone. In *Computing and Convergence Technology (ICCCT), 2012 7th International Conference on*, pages 763–766. IEEE, 2012.

[BR07] Baraniuk and Richard. Compressive sensing. *IEEE signal processing magazine*, 24(4):118–121, 2007.

[CCE+12] Wennan Chai, Cheng Chen, Ezzaldeen Edwan, Jieying Zhang, and Otmar Loffeld. 2d/3d indoor navigation based on multi-sensor assisted pedestrian navigation in wi-fi environments. In *Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS), 2012*, pages 1–7. IEEE, 2012.

[CCS07] Francesca Campolongo, Jessica Cariboni, and Andrea Saltelli. An effective screening design for sensitivity analysis of large models. *Environmental modelling and software*, 22(10):1509–1518, 2007.

[CDHH06] David Chu, Amol Deshpande, Joseph M Hellerstein, and Wei Hong. Approximate data collection in sensor networks using probabilistic models. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 48–48. IEEE, 2006.

[CEL+08] Andrew T Campbell, Shane B Eisenman, Nicholas D Lane, Emiliano Miluzzo, Ronald A Peterson, Hong Lu, Xiao Zheng, Mirco Musolesi, Kristóf Fodor, and Gahng-Seop Ahn. The rise of people-centric sensing. *Internet Computing, IEEE*, 12(4):12–21, 2008.

[CELea06] Andrew T Campbell, Shane B Eisenman, Nicholas D Lane, and et al. People-centric urban sensing. In *Proc. on Wireless internet 2006*, page 18. ACM, 2006.

[CFB+13] Giuseppe Cardone, Luca Foschini, Paolo Bellavista, Antonio Corradi, Cristian Borcea, Manoop Talasila, and Reza Curtmola. Fostering participaction in smart cities: a geo-social crowdsensing platform. *Communications Magazine, IEEE*, 51(6), 2013.

[CG13] Vladimir Coric and Marco Gruteser. Crowdsensing maps of on-street parking spaces. In *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*, pages 115–122. IEEE, 2013.

[CJZe09] Yukun Chen, Kai Jiang, Yu Zheng, and et.al. Trajectory simplification method for location-based social networking services. In *LBSN 2009*, pages 33–41, 2009.

[CKC13] Yohan Chon, Yunjong Kim, and Hojung Cha. Autonomous place naming system using opportunistic crowdsensing and knowledge from crowdsourcing. In *Proceedings of the 12th international conference on Information processing in sensor networks*, pages 19–30. ACM, 2013.

[CLK+13] Yohan Chon, Nicholas D Lane, Yunjong Kim, Feng Zhao, and Hojung Cha. Understanding the coverage and scalability of place-centric crowdsensing. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 3–12. ACM, 2013.

[CLLea11] David Chu, N. D. Lane, T. T. Lai, and et al. Balancing energy, latency and accuracy for mobile sensor data classification. In *ACM ENSS 2011*, pages 54–67, 2011.

[CLLea12] Yohan Chon, Nicholas D Lane, Fan Li, and et al. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proc. UbiComp 2012*, 2012.

[CLYea08] Jair Cervantesa, X. Lia, W. Yub, and et al. Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing*, 71(4-6):611–619, 2008.

[CWT06] Hu Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *VLDB Journal*, 15(3):211–228, 2006.

[DL14] Luke Dickens and Emil Lupu. On efficient meta-data collection for crowdsensing. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 62–67. IEEE, 2014.

[DP73] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. *The Canadian Cartographer*, 10:112–122, 1973.

[EBD+99] Christopher D Elvidge, Kimberly E Baugh, John B Dietz, Theodore Bland, Paul C Sutton, and Herbert W Kroehl. Radiance calibration of dmsp-ols low-light imaging data of human settlements. *Remote Sensing of Environment*, 68(1):77–88, 1999.

[FKK+11] Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 61–72. ACM, 2011.

[GKMea07] J. Gudmundsson, J. Katajainen, D. Merrick, and et al. Compressing spatio-temporal trajectories. In *ISAAC 2007*, pages 763–795, 2007.

[GM95] Robert A Gray and Peter S Maybeck. An integrated gps/ins/baro and radar altimeter system for aircraft precision approach landings. In *Aerospace and Electronics Conference, 1995. NAECON 1995., Proceedings of the IEEE 1995 National*, volume 1, pages 161–168. IEEE, 1995.

[GYL11] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *Communications Magazine, IEEE*, 49(11):32–39, 2011.

[GYZZ14] Bin Guo, Zhiwen Yu, Daqing Zhang, and Xingshe Zhou. From participatory sensing to mobile crowd sensing. *arXiv preprint arXiv:1401.3090*, 2014.

[HA13] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. Online, www.otexts.org, 2013.

[HCCL13] Xiping Hu, Terry HS Chu, Henry CB Chan, and Victor CM Leung. Vita: A crowdsensing-oriented mobile cyber physical system. 2013.

[HHD+12] Nengqiang He, Jinhai Huo, Yuhan Dong, Yipeng Li, Yang Yu, and Yong Ren. Atmospheric pressure-aware seamless 3-d localization and navigation for mobile internet devices. *Tsinghua Science and Technology*, 17(2):172–178, 2012.

[HKOS05] Rob J Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. Prediction intervals for exponential smoothing using two new classes of state space models. *Journal of Forecasting*, 24(1):17–37, 2005.

[HLZ+13] Xiping Hu, Qiang Liu, Chunsheng Zhu, Victor Leung, Terry HS Chu, and Henry CB Chan. A mobile crowdsensing system enhanced by cloud-based social networking services. In *Proceed-*

ings of the First International Workshop on Middleware for Cloud-enabled Sensing, page 3. ACM, 2013.

[HS92] John Hershberger and Jack Snoeyink. Speeding up the douglas-peucker line-simplification algorithm. In *Proc. 5th Intl. Symp. on Spatial Data Handling*, pages 134–143, 1992.

[HZY] Yang Han, Yanmin Zhu, and Jiadi Yu. A distributed utility-maximizing algorithm for data collection in mobile crowd sensing.

[IKI94] Mary Inaba, N. Katoh, and H. Imai. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In *Proc. ACM Symposium on Computational Geometry*, pages 332–339, 1994.

[Jen89] G.F. Jenks. Geographic logic in line generalisation. *Cartographica*, 26:27–42, 1989.

[JGEWE08] Shau-Shiun Jan, Demoz Gebre-Egziabher, Todd Walter, and Per Enge. Improving gps-based landing system performance using an empirical barometric altimeter confidence bound. *Aerospace and Electronic Systems, IEEE Transactions on*, 44(1):127–146, 2008.

[JHP05] G Jost, GBM Heuvelink, and A Papritz. Analysing the space–time distribution of soil water storage of a forest ecosystem using spatio-temporal kriging. *Geoderma*, 128(3):258–273, 2005.

[JKGS12] Yichao Jin, Parag Kulkarni, Sedat Gormus, and Mahesh Sooriyabandara. Content centric and load-balancing aware dynamic data aggregation in multihop wireless networks. In *IEEE WiMob 2012*, pages 179–186, 2012.

[Jol86] I. T. Jolliffe. *Principal component analysis*. Springer-Verlag, New York, 1986.

[Jon08] M. Prior Jones. Satellite communications systems buyer's guide. In *British Antarctic Survey*, 2008.

[JSS+12] Prem Prakash Jayaraman, Abhijat Sinha, Wanita Sherchan, Shonali Krishnaswamy, Arkady Zaslavsky, Pari Delir Haghighi, Seng Loke, and Manh Thang Do. Here-n-now: A framework for context-aware mobile crowdsensing. In *Proc. of the Tenth International Conference on Pervasive Computing*, 2012.

[Kan10] Eiman Kanjo. Noisespy: A real-time mobile phone platform for urban noise monitoring and mapping. *Mobile Networks and Applications*, 15(4):562–574, 2010.

[KH06] Elliott D. Kaplan and Christopher J. Hegarty. *Understanding GPS: principles and applications*. Artech House Inc., 2006.

[KHKZ08] Andreas Krause, Eric Horvitz, Aman Kansal, and Feng Zhao. Toward community sensing. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 481–492. IEEE Computer Society, 2008.

[KS03] JH Kim and S Sukkarieh. A baro-altimeter augmented ins/gps navigation system for an uninhabited aerial vehicle. In *Int. Symp. on Satellite Navigation Technologys*, 2003.

[Kup05] Axel Kupper. *Location-based services: Fundamentals and Operation*. John Wiley & Sons Ltd, 2005.

[KWS12] Friedrich Keller, Thomas Willemsen, and Harald Sternberg. Calibration of smartphones for the use in indoor navigation. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–8. IEEE, 2012.

[KXAA13] W Khan, Y Xiang, M Aalsalem, and Q Arshad. Mobile phone sensing systems: A survey. *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, 15(1):402–427, 2013.

[LCZ+13] Nicholas D Lane, Yohan Chon, Lin Zhou, Yongzhe Zhang, Fan Li, Dongwon Kim, Guanzhong Ding, Feng Zhao, and Hojung Cha. Piggyback crowdsensing (pcs): energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 7. ACM, 2013.

[LD12] Noam Levin and Yishai Duke. High spatial resolution nighttime light images for demographic and socio-economic studies. *Remote Sensing of Environment*, 119:1–10, 2012.

[LDR08] Ralph Lange, F. Drr, and K. Rothermel. Online trajectory data reduction using connection-preserving dead reckoning. In *Mobiquitous 2008*, 2008.

[LEM+08] Nicholas D Lane, Shane B Eisenman, Mirco Musolesi, Emiliano Miluzzo, and Andrew T Campbell. Urban sensing systems: opportunistic or participatory? In *Proceedings of the 9th workshop on Mobile computing systems and applications*, pages 11–16. ACM, 2008.

[LFDR09] Ralph Lange, Tobias Farrell, Frank Durr, and Kurt Rothermel. Remote real-time trajectory simplification. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1–10. IEEE, 2009.

[LHG13] Binghao Li, Bruce Harvey, and Thomas Gallagher. Using barometers to determine the height for indoor positioning. In *IGNSS Symposium 2013*, 2013.

[LMLe10] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, and et.al. A survey of mobile phone sensing. *IEEE communications magazine*, 48:140–150, 2010.

[LR04] Lixin Li and Peter Revesz. Interpolation methods for spatio-temporal geographic data. *Computers, Environment and Urban Systems*, 28(3):201–227, 2004.

[LRH11] Catherine T. Lawson, S. S. Ravi, and Jeong-Hyon Hwang. Compression and mining of gps trace data: New techniques and applications. Technical report, State University of New York at Albany, 2011.

[LTZG08] Dengsheng Lu, Hanqin Tian, Guomo Zhou, and Hongli Ge. Regional mapping of human settlements in southeastern china with multisensor remotely sensed data. *Remote Sensing of Environment*, 112(9):3668–3679, 2008.

[LXLea11] Nicholas D Lane, Ye Xu, Hong Lu, and et al. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proc. UbiComp 2011*, pages 355–364. ACM, 2011.

[LZD+12] Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, and Feng Zhao. A reliable and accurate indoor localization method using phone inertial sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 421–430. ACM, 2012.

[MLR12] Diego Mendez, M Labrador, and K Ramachandran. Data interpolation for participatory sensing systems. *Pervasive and Mobile Computing*, 2012.

[Mor91] Max D Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991.

[MPP11] Jonathan Muckell, Vikram Patil, and Fan Ping. Squish: An online approach for gps trajectory compression. In *Com.Geo 2011*, 2011.

[MPR08] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[MS99] AH Mohamed and KP Schwarz. Adaptive kalman filtering for ins/gps. *Journal of Geodesy*, 73(4):193–203, 1999.

[MSNS09] Nicolas Maisonneuve, Matthias Stevens, Maria E Niessen, and Luc Steels. Noisetube: Measuring and mapping noise pollution with mobile phones. In *Information Technologies in Environmental Engineering*, pages 215–228. Springer, 2009.

[NAS] NASA. U.s. standard atmosphere, 1976. Washington, U.S. Government Printing Office.

[Ney34] Jerzy Neyman. On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 97(4):558–625, 1934.

[NP82] J. Nievergelt and F. P. Preparata. Plane-sweep algorithms for intersecting geometric figures. *Communications of the ACM*, 25:739–747, 1982.

[NYZea12] Nima Nikzad, J. Yang, P. Zappi, and et al. Model-driven adaptive wireless sensing for environmental healthcare feedback systems. In *ICC2012*, pages 3489–3493, 2012.

[PBGS00] R Kelley Pace, Ronald Barry, Otis W Gilley, and CF Sirmans. A method for spatial–temporal forecasting with an application to real estate prices. *International Journal of Forecasting*, 16(2):229–246, 2000.

[PCL+11] Ling Pei, Ruizhi Chen, Jingbin Liu, Zhengjun Liu, Heidi Kuusniemi, Yuwei Chen, and Lingli Zhu. Sensor assisted 3d personal navigation on a smart phone in gps degraded environments. In *Geoinformatics, 2011 19th International Conference on*, pages 1–6. IEEE, 2011.

[PPS06] M. Potamias, K. Patroumpas, and T. Sellis. Sampling trajectory streams with spatio- temporal criteria. In *SSDBM 2006*, pages 275–284, 2006.

[PS06] Sang C. Parka and Hayong Shinb. Polygonal chain intersection. *Computers & graphics*, 26:341–350, 2006.

[PXGUS14] Layla Pournajaf, Li Xiong, Daniel A Garcia-Ulloa, and Vaidy Sunderam. A survey on privacy in mobile crowd sensing task management. Technical report, Technical Report TR-2014-002, Department of Mathematics and Computer Science, Emory University, 2014.

[RCMea12] Usman Raza, A. Camerra, A.L. Mruphy, and et al. What does model-driven data acquisition really achieve in wireless sensor networks? In *PerCom2012*, pages 85–94, 2012.

[RES10] Sasank Reddy, Deborah Estrin, and Mani Srivastava. Recruitment framework for participatory sensing data collections. *Pervasive Computing*, pages 138–155, 2010.

[RFA+13] Mirco Rossi, Sebastian Feese, Oliver Amft, Nils Braune, Sandro Martis, and G Troster. Ambientsense: A real-time ambient sound recognition system for smartphones. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 230–235. IEEE, 2013.

[Rip05] Brian D Ripley. *Spatial statistics*, volume 575. John Wiley & Sons, 2005.

[RLLPG12] Moo-Ryong Ra, Bin Liu, Tom F La Porta, and Ramesh Govindan. Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 337–350. ACM, 2012.

[SCP+11] Minho Shin, Cory Cornelius, Dan Peebles, Apu Kapadia, David Kotz, and Nikos Triandopoulos. Anonysense: A system for anonymous opportunistic sensing. *Pervasive and Mobile Computing*, 7(1):16–30, 2011.

[SDM10] Balaji Vasan Srinivasan, Ramani Duraiswami, and Raghu Murtugudde. Efficient kriging for real-time spatio-temporal interpolation. In *Proceedings of the 20th Conference on Probability and Statistics in the Atmospheric Sciences*, pages 228–235, 2010.

[SJK+12] Wanita Sherchan, Prem Prakash Jayaraman, Shonali Krishnaswamy, Arkady Zaslavsky, Seng Loke, and Abhijat Sinha. Using on-the-move mining for mobile crowdsensing. In *Mobile*

Data Management (MDM), 2012 IEEE 13th International Conference on, pages 115–124. IEEE, 2012.

[SM11] Tommy Szalapski and Sanjay Madria. Energy-efficient real-time data compression in wireless sensor networks. In *IEEE MDM 2011*, pages 236–245, 2011.

[SREB01] Paul Sutton, Dar Roberts, C Elvidge, and Kimberly Baugh. Census from heaven: an estimate of the global human population using night-time satellite imagery. *International Journal of Remote Sensing*, 22(16):3061–3076, 2001.

[SSR11] Rijurekha Sen, Pankaj Siriah, and Bhaskaran Raman. Roadsoundsense: Acoustic sensing based road congestion monitoring in developing regions. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2011 8th Annual IEEE Communications Society Conference on*, pages 125–133. IEEE, 2011.

[Ste95] James Stewart. *Multivariable calculus*. Cengage Learning, 1995.

[Sur05] James Surowiecki. *The wisdom of crowds*. Random House LLC, 2005.

[SZL10] Jing Shi, Yanchao Zhang, and Yunzhong Liu. Prisense: privacy-preserving data aggregation in people-centric urban sensing systems. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[Tay05] George Taylor. Line simplification algorithms. In *http://www.comp.glam.ac.uk/pages/staff/getaylor/papers/lcwin.pdf*, 2005.

[TBH13] Guliz S Tuncay, Giacomo Benincasa, and Ahmed Helmy. Participant recruitment and data collection framework for opportunistic sensing: a comparative analysis. In *Proceedings of the 8th ACM MobiCom workshop on Challenged networks*, pages 25–30. ACM, 2013.

[TCS+12] Andrei Tamilin, Iacopo Carreras, Emmanuel Ssebaggala, Alfonse Opira, and Nicola Conci. Context-aware mobile crowdsourcing. In *UbiComp*, pages 717–720, 2012.

[TLC09] Kang-Ting Tsai, Min-Der Lin, and Yen-Hua Chen. Noise mapping in urban environments: A taiwan study. *Applied Acoustics*, 70(7):964–972, 2009.

[VAA04] Mehmet C Vuran, Özgür B Akan, and Ian F Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245–259, 2004.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.

[WLW12] Song Wang, Juan Li, and Sihong Wang. A height controller design for miniature uavs based on gps and barometer. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 31–34. IEEE, 2012.

[WM03] S-T Wu and MR Gonzales Márquez. A non-self-intersection douglas-peucker algorithm. In *Computer Graphics and Image Processing, 2003*, pages 60–66, 2003.

[WSKe11] M. Wirz, P. Schlpfer, M.B. Kjrgaard, and et.al. Towards an online detection of pedestrian flocks in urban canyons by smoothed spatio-temporal clustering of gps trajectories. In *Proceedings of*

the 3rd ACM SIGSPATIAL International Workshop on LBSN, pages 17–24, 2011.

[WZX13] Leye Wang, Daqing Zhang, and Haoyi Xiong. effsense: energy-efficient and cost-effective data uploading in mobile crowdsensing. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 1075–1086. ACM, 2013.

[YAS03] Yutaka Yanagisawa, Junichi Akahani, and Tetsuji Satoh. Shape-based similarity query for trajectory of mobile objects. In *MDM2003*, pages 63–71, 2003.

[YKG10] Tingxin Yan, Vikas Kumar, and Deepak Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 77–90. ACM, 2010.

[YLWea11] Josh Ying, W. Lee, T. Weng, and et al. Semantic trajectory mining for location prediction. In *ACM GIS2011*, pages 34–43, 2011.

[YXFT12] 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*, pages 173–184. ACM, 2012.

[ZJK13] Arkady Zaslavsky, Prem Prakash Jayaraman, and Shonali Krishnaswamy. Sharelikescrowd: Mobile analytics for participatory sensing and crowd-sourcing applications. In *Data Engineering Workshops (ICDEW), 2013 IEEE 29th International Conference on*, pages 128–135. IEEE, 2013.

[ZLCea08] Y. Zheng, Q. Li, Y. Chen, and et al. Understanding mobility based on gps data. In *UbiComp 2008*, page 312321, 2008.

[ZPZO13] Jianwei Zhan, Jing Pang, Guozhu Zhang, and Gang Ou. Application of baro-altimeter sensor in emergency positioning and navigation based on compass geo satellites. *International Journal of Distributed Sensor Networks*, 2013, 2013.

[ZS09] Hao Zhenhai and Huang Shengguo. Data fusion approach based on high precision barometric altimeter and gps. In *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*, pages 1–4. IEEE, 2009.

[ZZXea09] Y. Zheng, L. Zhang, X. Xie, and et al. Mining interesting locations and travel sequences from gps trajectories. In *WWW 2009*, pages 791–800, 2009.