# Mobility sensing and analysis for urban transportation

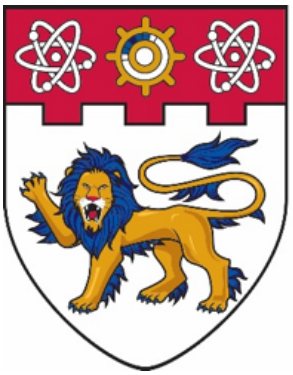
Tong, Panrong

2019

Tong, P. (2019). Mobility sensing and analysis for urban transportation. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/137310

https://doi.org/10.32657/10356/137310



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

# MOBILITY SENSING AND ANALYSIS FOR URBAN TRANSPORTATION

PANRONG TONG

**Interdisciplinary Graduate School**

**Nanyang Environment And Water Research Institute**

# MOBILITY SENSING AND ANALYSIS FOR URBAN TRANSPORTATION

**PANRONG TONG**

**Interdisciplinary Graduate School**
**Nanyang Environment And Water Research Institute**

A thesis submitted to Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

**2019**

# Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

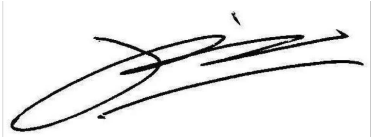
2020-03-15

Date

Panrong Tong

# Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.



2020-03-15

.....

Date

.....

Li Mo

# Authorship Attribution Statement

This thesis contains material from TWO paper(s) published in the following peer-reviewed journal(s) where I was the first and/or corresponding author.

Chapter 3 is published as W. Du, P. Tong, and M. Li. "UniLoc: A Unified Mobile Localization Framework Exploiting Scheme Diversity." In 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp. 818-829. IEEE, 2018.

The contributions of the co-authors are as follows:

- Dr. Mo Li provided the project direction and revised the manuscript drafts.
- I co-designed the study with Dr. W. Du and performed all the laboratory work at SCSE, NTU, Singapore. I also collected and analyzed the data generated in our experiments.
- Dr. W. Du prepared the manuscript drafts. The manuscript was revised by me and Dr. Mo Li .

Chapter 4 is published as P. Tong, W. Du, M. Li, J. H, W. Wang and Z. Qin. Last-mile School Shuttle Planning with Crowdsensed Student Trajectories. In IEEE Transactions on Intelligent Transportation Systems (2019), accepted to appear.

The contributions of the co-authors are as follows:

- Dr. M Li provided the project direction and technical direction. He also improved the writing quality of the paper. Dr. M. Li discussed with me at each stage and each procedure of this paper.
- I prepared the manuscript drafts. The manuscript was revised by Dr. W. Du, Dr. M. Li and Mr. J. H.

- I designed the study and performed all the laboratory work at SCSE, NTU, Singapore. I also analyzed the experimentally collected data.
- Dr. W. Wang and Dr. Z. Qin helped collect data and provided constructive advice during this work.

2020-03-15

Date

Panrong Tong

# Acknowledgment

First and foremost, I would like to thank my supervisor, Dr. Mo Li, for the patient guidance, encouragement and advice he has provided throughout my PhD years. When I joined WANDS, I got no experience on doing research and lacked the knowledge base to conduct cutting-edge research. I feel lucky to have a supervisor who sees my potentials, cares for my personal development and supports me with insightful knowledge and professional skills. Dr Li is also a helpful friend in my personal life. I could not imagine having a better supervisor for my PhD study.

I would like to express gratitude to my co-supervisor Dr. Zhiwei Wang and mentor Dr. Law Wing-Keung, for their support and guidance. Their insightful advice helps me realize the importance of being interdisciplinary when doing research.

I want to thank my collaborator, Dr. Wan Du, for his help in my research projects. His insightful opinions would always lead me to the correct direction when I met problem in my research projects. His valuable comments help to greatly enhance the quality and readability of the writing. It is my great honor to work with Dr. Wan Du.

I am also grateful to my friends in WANDS: Yuanqing Zheng, Zhenjiang Li, Wan Du, Zhidan Liu, Pengfei Zhou, Jiajue Ou, Yuxiao Hou, Shiqi Jiang, Jansen Christian Liando, Yaxiong Xie, Shuyu Shi, Chu Cao, Xiaoyun Mo, Weiping Sun, Jinlong E, Sijie Ji, Yanbo Zhang, Agustinus Wellson Tengourtius, Amalinda Gamage, Mingqian Li.

Last but not least, I would like to thank my family. It is your love and support that help me complete my PhD study.

# Contents

**Statement of Originality** **i**

**Supervisor Declaration Statement** **ii**

**Authorship Attribution Statement** **iii**

**Acknowledgement** **v**

**List of Acronyms** **ix**

**Lists of Figures** **xi**

**Lists of Tables** **xii**

**Abstract** **xiii**

**1 Introduction** **1**

1.1 UniLoc: Exploiting the Diversity of Localization Scheme for Improved individual mobility sensing . . . . . 3

1.2 Last-mile School Shuttle Planning with Crowdsensed Student Trajectories 7

1.3 Large-Scale Vehicle Trajectory Reconstruction with Camera Sensing Network . . . . . 10

1.4 Organization of This Thesis . . . . . 12

**2 Related Work** **14**

2.1 Model aggregation . . . . . 14

2.2 Localization error . . . . . 14

2.3 Individual localization schemes . . . 15
2.4 Bus stop selection (BSS) . . . 16
2.5 Bus route generation (BRG) . . . 17
2.6 Common practices in school bus planning . . . 17
2.7 Data-driven bus route planning . . . 18
2.8 Travel mode detection . . . 18
2.9 Path inference . . . 19
2.10 Vehicle re-identification (Re-ID) . . . 19
2.11 Multi-camera tracking . . . 21

**3 UniLoc: Exploiting the Diversity of Localization Scheme for Improved Individual Mobility Sensing 22**
3.1 Motivation . . . 22
3.2 Error modeling . . . 24
3.2.1 General workflow . . . 25
3.2.2 Error models . . . 26
3.3 UniLoc . . . 31
3.3.1 UniLoc1: selecting the "best" localization scheme . . . 31
3.3.2 UniLoc2: locally-weighted BMA-based localization . . . 31
3.3.3 Gain of locally-weighted BMA localization . . . 33
3.3.4 Energy consumption . . . 34
3.4 Evaluation . . . 35
3.4.1 Experiment setting . . . 35
3.4.2 Error model validation . . . 35
3.4.3 Accuracy . . . 36
3.4.4 Energy consumption . . . 41
3.4.5 Response time . . . 43

**4 Last-mile School Shuttle Planning with Crowdsensed Student Trajectories 44**
4.1 Motivation . . . 44
4.1.1 The National Science Experiment (NSE) . . . 44
4.1.2 Challenges . . . 46

4.2 Design . . . 47
4.2.1 Trajectory Profiling . . . 48
4.2.2 Graph Construction . . . 52
4.2.3 Graph-based Route Planning . . . 57
4.3 Evaluation . . . 61
4.3.1 Experiment settings . . . 61
4.3.2 Mobility Statistics of NSE Trajectories . . . 62
4.3.3 System performance . . . 63
4.3.4 System Components . . . 66

**5 Large-Scale Vehicle Trajectory Reconstruction with Camera Sensing Network 69**
5.1 Sensing with Cameras . . . 69
5.1.1 Why Vehicle Trajectories? . . . 69
5.1.2 Camera Sensing Network . . . 70
5.1.3 Uncertainties in Vehicle Identification . . . 73
5.2 VeTrac Design . . . 77
5.2.1 Multi-Dimensional Similarity (MDS) . . . 77
5.2.2 Learning Vehicle Identities with Graph . . . 80
5.2.3 Self-Training with Road Network Graph . . . 85
5.3 Evaluation . . . 87
5.3.1 Experimental Setups . . . 87
5.3.2 Experimental Results . . . 89
5.3.3 Component Performance . . . 94

**6 Conclusion and Future Work 96**
6.1 Conclusion . . . 96
6.2 Future Work . . . 98

# List of Acronyms

| | |
|---|---|
| **RSSI** | Received Signal Strength Indicator |
| **PDR** | Pedestrian Dead Reckoning |
| **BMA** | Bayesian Model Averaging |
| **VRP** | Vehicle Routing Problem |
| **HMM** | Hidden Markov Model |
| **NSE** | National Science Experiment |
| **IMU** | Inertial Measurement Unit |
| **GPS** | Global Positioning System |
| **API** | Application Program Interface |
| **OSM** | OpenStreeMap |
| **CNN** | Convolutional neural network |
| **LPR** | License Plate Recognition |
| **Re-ID** | Re-identification |

# List of Figures


1.1 UniLoc framework. . . . . 5
1.2 A path from an office to a restaurant. The numbers are the distances from the start point to that location. . . . . 6
1.3 Localization error of different schemes along a daily path on our campus.At each location, as we know the ground truth in the experiment, Skyline chooses the best scheme as its result. . . . . 6
1.4 Multiple pickup locations for each student . . . . . 9

3.1 Office layout plan (56×20 $m^2$) and offline fingerprints. . . . . 27
3.2 The user's true location is at (0, 0.3). Three localization schemes provide three estimated locations (S1, S2 and S3). In two examples, each scheme is assigned with different weights. The weights (w) and localization results (U1 or U2) of two examples are highlighted by different colors (red or blue). The localization errors of UniLoc1 and UniLoc2 are shown with dash and solid lines respectively. . . . . 32
3.3 Eight daily paths on our campus. . . . . 37
3.4 Localization error on the eight daily paths. . . . . 38
3.5 Localization error of the optimal single-selection solution and UniLoc along the daily path 1. . . . . 38
3.6 Localization error for all underlying schemes and UniLoc in different places or with heterogeneous devices. . . . . 39
3.7 Evaluations on the daily path 1. . . . . 40

4.1 NSE Trajectories Spatial Distribution . . . . . 45
4.2 Imperfection of NSE data . . . . . 47
4.3 System overview . . . . . 48

4.4 Routes suggested by Google Directions Service. . . . . 51
4.5 The construction of the route graph . . . . . 52
4.6 Examples of supported operators in proposed data structure $G_{route}$ . . . 54
4.7 Visualization of the real route graph . . . . . 55
4.8 The shortsightedness of greedy expansion . . . . . 58
4.9 NSE Mobility Statistics . . . . . 62
4.10 Visualization of last-mile bus routes planned for School A . . . . . 65
4.11 Overall performance . . . . . 66
4.12 Evaluation on the graph structure and Tabu-based expansion algorithm . 66

5.1 Deployment schemes and example frames of two common cameras in cities 70
5.2 Camera distribution in urban area . . . . . 71
5.3 Statistic description . . . . . 72
5.4 Identity uncertainties of vision-based solutions . . . . . 74
5.5 The Impact of identity uncertainties for LPR and Re-ID solutions . . . . 75
5.6 Multi-dimensional Similarity (MDS) block . . . . . 80
5.7 VeTrac: System overview . . . . . 81
5.8 The proposed two-layer GCN units . . . . . 83
5.9 Highway: visualization of origins and destinations . . . . . 90
5.10 Highway: An representative example. The black crosses in (b) and (c) denoted the locations of the above seven snapshots in the representation spaces. . . . . 91
5.11 Highway: statistical evaluation . . . . . 92
5.12 Robustness evaluation . . . . . 92
5.13 Urban: comparison of reconstructed paths . . . . . 93
5.14 Urban: statistical evaluation . . . . . 94
5.15 The effectiveness of MDS block . . . . . 95
5.16 The effectiveness of self-training . . . . . 95

# List of Tables


3.1 Influence factors of typical localization models. . . . . 26
3.2 Estimated coefficients for four typical localization schemes, i.e., RADAR [2], the cellular-based localization scheme [88], the motion-based localization scheme [64] and Travi-Navi [141]. . . . . 26
3.3 Normalized RMSE of the online error prediction for different localization schemes. . . . . 36
3.4 Power and energy consumption of UniLoc and all localization schemes along the daily path 1. . . . . 42
3.5 Average response time for one location estimation, including computation and transmission. . . . . 42

4.1 Success rate of real route extraction by Google Directions Service. The success rate of the Google path set indicates the probability that the set contains the real path traversed by a student. The success rate of the heuristic selection methods indicates the likelihood that the method outputs the real path. . . . . 50
4.2 Skeleton of the route graph . . . . . 53
4.3 Summary of student commute patterns . . . . . 64
4.4 Summary of overall performance . . . . . 64
4.5 Comparison of trip identification algorithms . . . . . 67
4.6 The accuracy of travel mode detection . . . . . 68
4.7 The performance of pickup location extraction . . . . . 68

5.1 A record of vehicle snapshot database . . . . . 72

# Abstract

Understanding urban mobility is becoming one of the most critical tasks as more of the world's cities become crowded and congested. Given the increasing volume and variety of urban mobility, traditional survey based methods are insufficient to sense the whole picture, based on which few need-satisfying solutions can be built. Fortunately, over the past few years, we also experienced a significant increase in peoples ability to collect data from various sensors, smartphones, or other devices, in different formats. During my PhD program, I focus on designing holistic systems for sensing and analyzing mobility for urban transportation. I aim at mining insightful mobility information from not only dedicated sensors but also from smartphones, wearable devices and city infrastructures like traffic cameras. In this thesis, I present three studies for mobility sensing and analysis of different subjects and scale in urban context.

The first study considers the mobility of individual smartphone user. I observed that although various algorithms have been proposed to infer a user's location from data sensed by the smartphone, none of existing localization algorithms can works accurately in all environment, which results in incomplete or noisy mobility. To address this problem, I propose UniLoc, a unified framework that exploits the scheme diversity to gain extra performance improvement from state-of-the-art localization solutions. In UniLoc, multiple localization schemes are executed in parallel and independently. UniLoc predicts the localization accuracy of each scheme online according to the real-time context at per-location granularity. In a simple version, UniLoc chooses the best scheme as its final result at every location. Beyond that, UniLoc combines the results of all available schemes based on a locally-weighted Bayesian model averaging algorithm. The combined result is better than any individual scheme. Extensive experiments demonstrate that such an easy aggregation incurs little overhead in energy consumption or training a new scheme, but gains substantially from the localization scheme diversity.

The second study considers mobility of thousands of students. I found that collective mobility information can be crowdsensed from wearable devices and better urban services can thus be planed via proper analysis. By processing a large dataset composed of daily trajectories of thousands of students in Singapore, I find that, instead of simply picking up students from their homes, an optimal school shuttle planning system needs to learn the real transportation usage and plan across all potential pickup locations for every student to generate need-satisfying routes. It is challenging, however, to perform route planning over a large number of students each having multiple potential pickup locations. We develop a graph-based data structure that embeds potential pickup locations of all students with the awareness of real-world constraints and existing public transits. Based on the graph structure, we prove that the optimal last-mile school shuttle planning problem is NP-hard and thereafter design a Tabu-based expansion algorithm to solve the problem, which strikes at a proper balance between the savings of students' commute time and the total cost of operating the shuttle buses. Extensive experiments with large-scale real-world crowdsensed trajectory data demonstrate that our last-mile school shuttles can save the traveling time for most students by over 20% and the savings can be up to 65% for 10% of the students.

The third study considers the mobility of all general vehicles in a city. Vehicle trajectories provide fundamental understanding of the urban mobility and benefit a wide range of urban applications. However, state-of-the-art solutions for vehicle sensing (i.e., tracking with in-vehicle devices, roadway monitoring with flow sensors) can only acquire either partial or anonymized observations, thus fails to reconstruct all individual vehicle trajectories. To address those drawbacks, We propose VeTrac, a comprehensive system that uses widely deployed traffic cameras as a sensing network and reconstructs the complete knowledge of all general vehicle in the city. Despite recent advances in identifying vehicles based on their plates or appearance, we observe substantive vehicle identity uncertainties from existing methods. Accordingly, we design VeTrac that exploits the multi-dimensional similarity and embeds that in a graph convolution process in order to reduce the uncertainties of vehicle identification. VeTrac also employs a self-training process that incorporates the complex mobility dependency among the urban road networks to improve accuracy. Extensive experiments demonstrate that VeTrac reconstructs trajectories with an overall 89% accuracy and outperforms existing algorithms.

# Chapter 1

# Introduction

Understanding urban mobility is becoming one of the most critical tasks as more of the world's cities become crowded and congested. A comprehensive study on urban mobility [113] points out two global trends: (i) Mobility demands will continue to increase due to the urban population explosion. By 2050, 67% of the world's population are expected to live in urban environment, up from about 50% today. Nowadays, we already have 64% of all travel kilometers made in urban cities and the total number of kilometers traveled in cities are likely to tripe by 2050. (ii) Mobility demands are evolving towards complexity and most of the cities still have significant potential for improvement in addressing those needs. On the one hand, with the growth of transportation networks and their increasing interconnections, multimodal trips - which are trips comprising two or more transportation modes - are becoming a common travel phenomenon in urban areas. On the other hand, a study [63] evaluates the mobility performance of 84 cities worldwide and reveals that, on average, these 84 cities achieve less than half of the potential that could been reached by understanding and handling their mobility needs more wisely. Given the increasing volume and variety of urban mobility, traditional survey based methods usually are insufficient to sense the whole picture, based on which few need-satisfying solutions can be built.

Fortunately, over the past few years, we also experienced a significant increase in people's ability to collect data from various sensors, smartphones, or other devices, in different formats. For sensing with dedicated sensors, wireless sensor networks, as a traditional but efficient way to sense the environments, continue to be key solutions to

a lot of sensing applications in various fields. In the meantime, thanks to the advances in hardware industry, a lot of mobile devices, for example smartphones, have embedded with multiple accurate sensors of different types. Those devices are already capable of acting as sensory gateways to collect real-time data on people from all walks of life and environments of all different kinds. With the wide availability of informative data, crowdsourcing, which originally refers to the process of obtaining needed services, ideas, or content by soliciting contributions from a large group of people via websites, can be extended to solve challenges in sensing urban mobility.

During my PhD study, I focus on designing holistic systems for sensing and analyzing mobility for urban transportation. We aim at mining insightful mobility information from not only dedicated sensors but also from smartphones, wearable devices and transportation infrastructures like traffic cameras in a collective way. Those devices and infrastructures can serve as powerful sensing networks to observe fine-grained mobility pattern in a large scale, where both the granularity and coverage are difficult to be achieved by traditional survey based sensing even at the cost of massive efforts. However, data obtained in this way are usually imperfect, for example, data from different sensors may conflicts requiring unearthing more reliable observations, crowdsensed trajectories typically have large location drifts and could suffer from sparsity, and sometimes the data is not structured and information are hidden in videos and photos. To obtained useful insights that improve the efficiency of existing transportation systems, proper system design is needed. In this thesis, we present three works for mobility sensing and analysis of different subjects and scale in urban context.

- Mobility of individual smartphone user. In this project, we observed that although various algorithms have been proposed to infer a user's location from data sensed by the smartphone, none of existing localization algorithms can work accurately in all environment, which results in incompleted or noisy mobility. To address this problem, we propose a novel algorithm that leverages different sensors in smartphone to better determine users locations in various environment.

- Mobility of thousands of students. In this project, we exploited a more cost-efficient way of shuttles by letting the school bus operating near the students possible exits of public transits (which we learn from crowdsensed trajectories). Such a service complements public transits and provides last-mile extension of existing public

transportation for students. Different from traditional bus design that relies on origin-destination surveys to estimate travel demands, we leverage crowdsensed trajectories from students' wearable devices. A planning system is designed to learn the real public transportation usage and true travel demands from imperfect crowdsensed trajectories, and then plans need-satisfying routes based on that.

- Mobility of all vehicles in a city. In this project, we utilize densely deployed traffic cameras as a sensor network to profile trajectories of individual vehicles in a city. Different from onboard devices based solutions or roadway sensors based solutions, crowdsensed traffic camera videos record all vehicles passed with their identity cues. However, this information is not structured. Moreover, since traffic cameras are deployed for manual observation instead of vehicle path reconstruction, the identify information contains uncertainty. Thus this project presents VePath, a vehicle path reconstruction system that leverage multi-dimensional similarities and graph propagation based modeling to reconstruct individual vehicle paths from low-resolution videos.

## 1.1 UniLoc: Exploiting the Diversity of Localization Scheme for Improved individual mobility sensing

In the past decade, various localization schemes [2, 16, 17, 20, 24, 33, 39, 46, 48, 60, 64, 65, 69, 70, 76, 83, 88–90, 96, 104, 109, 111, 115, 124, 127, 128, 141, 145] have been developed for mobile devices using different sensors, e.g., GPS, Wi-Fi RSSIs (Received Signal Strength Indicator) [2, 16, 17, 65, 83, 128], and inertial sensors (i.e., accelerometer, gyroscope and magnetometer) [20, 48, 64, 96, 115, 127, 141]. However, as sensor data quality[1] changes according to environmental conditions, the performance of an individual scheme varies spatiotemporally [64]. To provide more stable performance, some works fuse the raw data of multiple sensors, e.g., Wi-Fi RSSIs are used to refine the results of motion-based Pedestrian Dead Reckoning (PDR) by particle filtering [48, 115, 141]. With predefined types of sensors, it is hard for a fixed fusion algorithm to automatically adapt to all possible environmental conditions. For instance, in some regions with weak Wi-Fi signals (due to high wireless interference or sparse deployment of access points), Wi-Fi RSSIs

may not be able to help the default motion-based PDR, or even make the estimated location depart from the user's true location. As a consequence, there does not yet exist a one-size-fit-all mobile localization system that can cover all the places in people's daily life.

In this study, we propose *UniLoc*, a unified framework that exploits the diversity of existing localization schemes to achieve accurate and robust positioning across variant environment. Different localization schemes use different sensors and they may provide complementary information for estimating the user's location. UniLoc adopts a novel fusion methodology that without going into the details of individual schemes, only processes the final outputs to exploit the complementary information of all available schemes and provide better result than any individual schemes. Based on such a design principle, UniLoc provides three features.

- **General**. UniLoc is not constrained to any specific localization schemes or sensor data. Any localization scheme can be easily integrated into UniLoc.
- **Adaptive**. UniLoc can automatically adapt to the spatiotemporal variation of sensor data at every location.
- **Scalable**. UniLoc can be used in new unknown places without pre-training.

It is challenging to transform such a framework into a practical system. First, we need an online error prediction method that can estimate the localization error of any schemes (*general*) at every location (*adaptive*) and in any places (*scalable*). Although some error models have been proposed [22,26,69], they are dedicated to special localization schemes and do not consider the real-time context either, but just assign a constant accuracy level to a scheme in an entire place. Second, it is hard to integrate the complementary information of multiple schemes only based on their final outputs without knowing the details of specific localization algorithms. Moreover, such an integration should also adapt to real-time context.

Our key observation is to tackle the above challenges from the perspective of sensor data. The localization error at one location is only determined by the specific localization algorithm and the instant sensor data. All factors (e.g., sensor specifications and

[1]In this study, we refer the quality of sensor data as its capacity in labeling and distinguishing different locations. It is determined by both sensor specifications and instant environmental conditions, e.g., Wi-Fi AP (Access Point) deployment or visible GPS satellites.

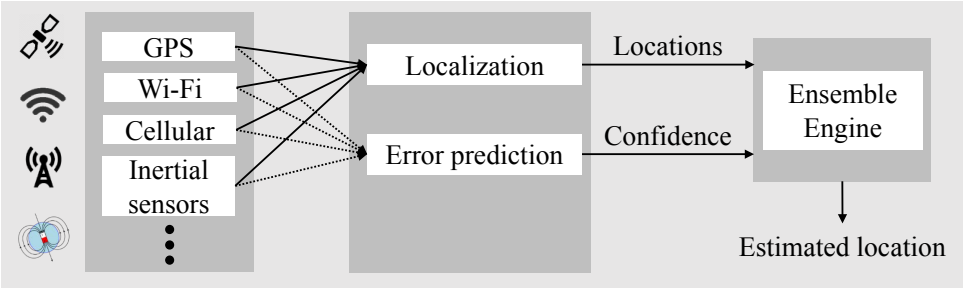



Figure 1.1: UniLoc framework.

environmental conditions) that implicitly impact the localization accuracy take effect by changing the instant sensor readings. We find some potential data features for each sensor type. We then can quantify the deterministic relation between the localization error and sensor data quality by training a regression model. In such a way, the implicit influence factors are captured by a set of explicit data features, and the regression model of one localization scheme is consistent at different places and *scalable* to new places without training. The confidence in each localization scheme's output can be calculated online according to the real-time sensor data at every location *adaptively*. Furthermore, since the data features are quantified directly by sensor data, we do not require the details of specific localization algorithms. The outputs of all available schemes are combined using an ensemble learning algorithm. Any localization schemes can be easily added into our *general* framework. For example, sensor data fusion based localization schemes, like Travi-Navi [141] and UnLoc [115], can also be treated as an individual scheme in our framework.

In UniLoc, as depicted in Figure 1.1, multiple *localization schemes* run in parallel. An *error model* is used to online estimate the accuracy of each scheme according to the real-time context. A probabilistic confidence in the output of each localization scheme is calculated. Both the output and confidence of all available schemes are processed in an *ensemble engine*. UniLoc combines the outputs of all schemes using a locally-weighted Bayesian Model Averaging (BMA) algorithm. Our work differs fundamentally from previous BMA-based approaches [46] in that we adapt the weight of each scheme at every location according to the real-time context. In UniLoc, the optimal weight of each scheme is approached by the confidence in its result at every location. Such a weighted averaging can better tolerate the uncertainty in online error prediction, and the

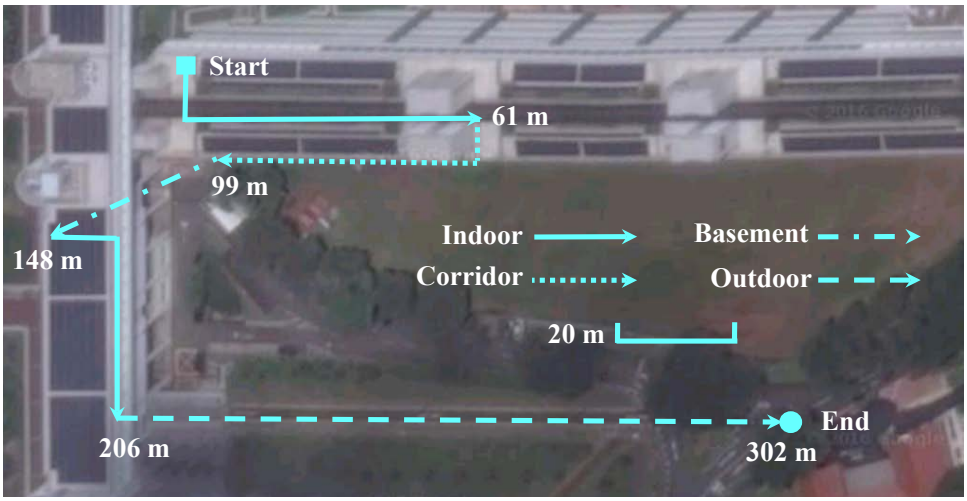



Figure 1.2: A path from an office to a restaurant. The numbers are the distances from the start point to that location.

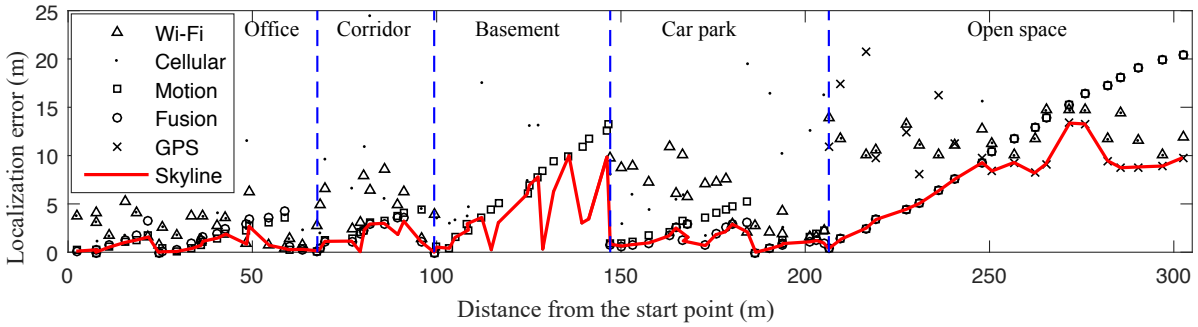



Figure 1.3: Localization error of different schemes along a daily path on our campus.At each location, as we know the ground truth in the experiment, Skyline chooses the best scheme as its result.

combined result is more accurate than the output of any individual schemes. In addition, the locally-weighted BMA-based algorithm offers UniLoc high flexibility. UniLoc can temporarily exclude one localization scheme by simply setting its confidence as zero, if it is not available in some regions, e.g., no Wi-Fi signal.

The computation of UniLoc, including both error modeling and BMA-based averaging, is light-weight, as they only involve simple linear calculation. To make UniLoc a general framework that can integrate some energy-consuming localization schemes, two techniques are adopted, i.e., GPS downsampling and offloading. First, GPS is turned off when its error is predicted to be large at some locations. Second, to avoid some localization schemes consuming much energy on smartphones, we move their computation to a

server. By intelligently processing some raw sensor data on smartphones, the transmitted data and the energy consumed by data transmissions are both minimized. Even with five schemes running in parallel, according to our experiments, UniLoc only increases the energy consumption of the most energy-efficient scheme (i.e., the motion-based PDR) by 14%.

Our experience shows that instead of studying new localization algorithms, easy aggregation of the end results from state-of-the-art schemes can already gain substantial extra performance improvement. Such experience indicates stretched design space considering the enormous availability of existing localization solutions. We implement a UniLoc framework that aggregates five existing localization schemes. We evaluate its performance in a variety of environments. Most of the experiments, >89%, are conducted in new places where we did not conduct any experiments to train the error models. The experiments demonstrate that UniLoc incurs little overhead in energy consumption and model training, but gains substantial performance improvement from scheme diversity (i.e., 1.6× error reduction against individual localization schemes).

In this study, we make the following contributions.

- We experimentally study the diversity of existing localization schemes.
- We design and implement a unified localization framework that features an online error prediction approach and a locally-weighted BMA localization algorithm.
- We validate the error models of five localization schemes and reveal the significant gain of the scheme diversity substantiated by UniLoc with extensive experiments.

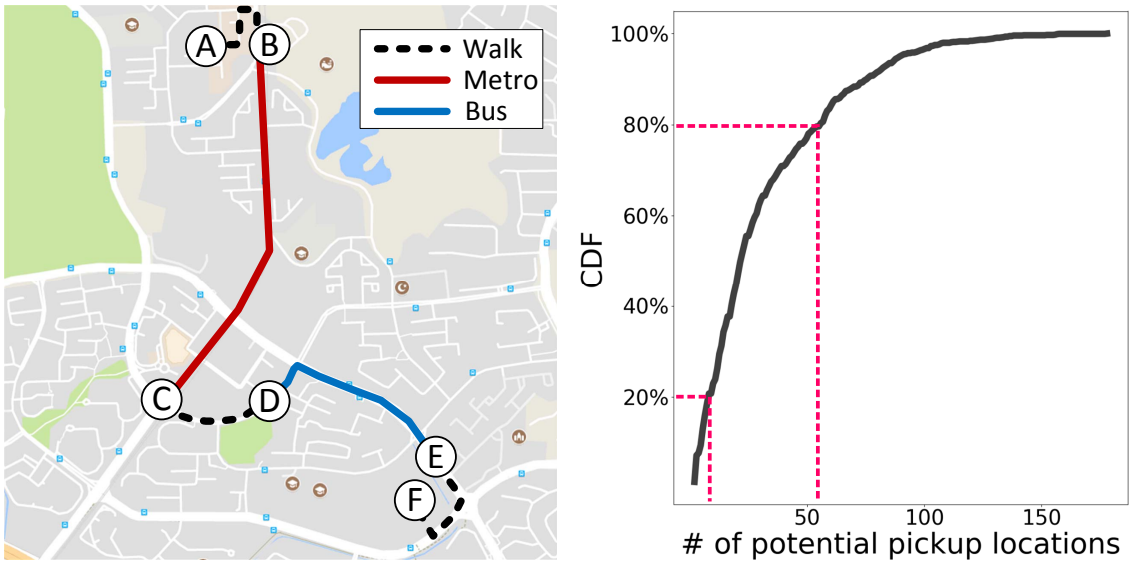
## 1.2 Last-mile School Shuttle Planning with Crowd-sensed Student Trajectories

Last-mile shuttles have become critical for urban transportation in modern cities, since they complete last legs of individual trips by getting people from transportation networks to their final destinations (usually where public transportation does not reach) [62]. Intuitively, providing such a service to students is both beneficial and low-cost due to their common destinations (i.e., the schools) and commuting hours. However, planning a need-satisfying last-mile school shuttle service involves two key tasks, i.e., estimating

transport demands (locations where users may get on the shuttles) and optimizing bus routes according to the estimated demands [4].

Existing practices of school shuttle planning heavily relies on offline surveys or empirical experience to estimate transport demands, which may not be accurate and is often inefficient. They either completely ignore the public transportation by assuming all the transport demands start from students' homes(e.g., the door-to-door school shuttles, which pick up students directly from their homes), or have simplified approximation on how students utilizing the public transportation (e.g., the metro school shuttles, which assumes most of the students take metro to one or few major stations nearest to their school and picks up them from the one or few stations). Some recent data-driven studies learn the true transport demands from personal mobility data (e.g., cellular footprints or taxi trips) [14, 18, 75, 132]. With that, they formulate the route planning problem into classic optimization problems such as vehicle routing problem (VRP) [42]. For example, Feeder [132], a most recent work, plans a last-mile shuttle route that takes commuters from a metro station to their destinations. With the cellular data of mobile users, Feeder learns the rough locations of their destinations, clusters them as bus stops, and plans routes accordingly.

However, the above data-driven approaches are limited by the granularity of their observations and thus often oversimplify the true transport demands for the following two reasons. First, previous works [14, 18, 75, 132] implicitly assume only one potential pickup location for each individual, which is often not the case in practice. Given the multiple choices of transportation (e.g., bus, metro, walk) and their combinations, potential pickup locations of a student could include the home, the entries and exits of used public transits (e.g., bus stops, metro stations) and all the road segments walked. For example, Figure 1.4a shows a representative home-school trip that contains 23 potential pickup locations, including home (point A), metro stops (point B and C), bus stops (point D and E) and all the road segments traversed by walk (dash lines). According to our observation from the crowdsensed trajectory data, most students have 6 to 52 potential pickup locations, as summarized in Figure 1.4b. Second, previous works [14, 18, 75, 132] are based on simple proximity model which is unaware of real-world constraints and existing public transits. For example, simply clustering geographically proximate transport demands and assigning one bus stop to serve all of them has been widely used in previous works, which




(a) A representative trip (b) Pickup locations of NSE trips

Figure 1.4: Multiple pickup locations for each student

however is not accurate because walking between geographically proximate locations may incur high cost due to road constraints (e.g., crossing a street but from a far away pedestrian flyover). Such unawareness of constraints or conveniences leads to suboptimal bus route planning.

This study presents last-mile school shuttle planning system that considers multiple pickup locations for each student with real-world constraints and existing public transits. We study daily trajectories crowdsensed from 2809 students through a nation-wide experiment of Singapore. The trajectories contain students' periodical locations and activity updates when they commute between home and school. The granularity and scale of our data allow us to finer profile their trips, fully examine the real transportation usage and generate more truthful demands to the shuttle service. Data samples in our trajectories are usually subject to a large location drift and may suffer from sparsity in some regions (see in Section 4.1.2). We thus proposes a trip profiling scheme to infer precise traveling paths and transportation modes, and thus extract potential pickup locations for all students.

With multiple pickup locations for every student, we gain an extra dimension of optimization and can thus derive better shuttle route plans. However, it is computational infeasible to extend existing VRP based algorithms to process such scale of demands.

For 500 students and each having 20 potential pickup locations, blindly applying existing algorithms would result in the steps of selecting one possible pickup location for each student, and then running the algorithms once for each one out of $20^{500}$ possible combinations. Thus, this study further proposes a novel graph-based data structure that embeds all transport demands on the road network. Such a graph based data structure aggregates similar demands from different students and provides a set of operators that facilitate route plan and update.

With the proposed data structure, we thus develop a customized Tabu expansion algorithm to find a proper subset of nodes in the graph as bus stops and lay the bus routes. The proposed solution is able to balance the commute time saved for all students and the operating cost of the shuttle buses.

We evaluate the performance of our last-mile school shuttle routes with the trajectories of 2809 students from 7 schools in Singapore. According to the evaluation results, our solution is able to save the commute time of most students by over 20%, where 10% of the students can save up to 65% while 75% of the students can save at least over 8%. In summary, this study makes the following contributions:

- This is the first study, to the best of our knowledge, to demonstrate the necessity of considering multiple pickup locations of each individual for more efficient shuttle bus planning.
- This study develops a holistic last-mile bus planning system with a set of techniques, including trip profiling from trajectory data, a novel graph-based data structure for embedding travel demands, as well as a graph-based Tabu expansion algorithm.
- Extensive experiments are performed on real-world crowdsensed data to evaluate the proposed system and compare with benchmark solutions.

## 1.3 Large-Scale Vehicle Trajectory Reconstruction with Camera Sensing Network

Knowing the moving trajectories of millions of vehicles traveling in a city provides fundamental understanding of the urban mobility, and the reconstruction of such knowledge benefits a wide range of applications in transportation and urban redevelopment, e.g.,

usage-based electronic toll collection (ETC), traffic estimation and prediction, intelligent traffic light control, origin-destination (OD) analysis, planning of functional regions within the city, and many others.

State-of-the-art solutions to trajectory reconstruction are mainly based on (i) tracking with in-vehicle devices (e.g., GPS), which however only obtains partial knowledge from limited vehicle fleets, and (ii) roadway monitoring with flow sensors (e.g., loop detectors, piezoelectric sensors, and infrared sensors), which however only captures traffic flows, i.e., the amount of vehicles passing by road segments, thus not applicable to reconstructing individual vehicle trajectories.

In this study, we study the use of widely deployed traffic cameras as a sensing network to observe general vehicle mobility and based on that reconstruct the complete knowledge of all general vehicles in the city with high accuracy. Compared with existing solutions, the use of the traffic camera network possesses the following advantages: (i) Traffic cameras are generally available in urban cities and requires no additional deployment cost; (ii) Traffic cameras are widely deployed and provide high coverage of the city for vehicle sensing and monitoring; (iii) Advanced video and image analysis techniques can be applied to extract vehicle identifies which provides essential information for recovering their trajectories.

With the vehicle identities being discovered from traffic cameras along their moving trajectories, a straightforward solution is to maintain such identities across all roadways and reconstruct the trajectory of each vehicle by concatenating its sequence of time-stamped observations from all concerned traffic cameras. Applying such a solution in practice is challenged by the inherent incompleteness and inaccuracy of the camera sensing network. The traffic camera observations are spatial-temporally incomplete - not all road junctions are covered by traffic cameras and that results in uncertainties in inferring the true moving trajectories over those uncovered areas. Such data incompleteness is made more challenging when considering camera malfunctions and data loss. The vehicle identities extracted from video and image analysis may be inaccurate, and it is further complicated due to the heterogeneity among traffic cameras and their data quality - the vehicle snapshots taken from different cameras (e.g., front-view v.s. gear-view) and under different environment conditions are of different quality in resolution, view field, focus control, and clarity. State-of-the-art vehicle identification based on LPR (License Plate

Recognition) may give inaccurate results for low-quality camera data.

This study proposes a comprehensive system design centered around reducing the uncertainties of vehicle identification. The proposed design considers the vehicle identities based on both their appearance in the camera data as well as their mobility dependency across locations where those camera data are obtained. Multi-dimensional similarity examination is performed with vehicle snapshots taken from different traffic cameras. Specifically, our system examines similarities on LPR text, vehicle appearance, and mobility causality across the locations of observation, and gains combined confidence to relate vehicle identities across different camera observations. A graph convolution network (GCN) model is built to cluster vehicle snapshots, and based on that maintain the identity consistency across different camera observations. With GCN, our approach benefits from its efficacy in belief propagation and is able to converge to accurate clusters of representations. Finally our system joins the graph of GCN and that of the urban road network, and undergoes a self-learning process which incorporates the complex mobility dependency among the urban road network into the GCN, and thus further improves accuracy while recovering the individual vehicle trajectories.

We implement the proposed approaches and build an end-to-end system - VeTrac. We extensively experiment VeTrac with real-world data collected from a network of 1342 cameras deployed in an urban region. A full day camera data consisting of over 7 million vehicle snapshots are examined. We evaluate VeTrac performance based on a set of manually labeled vehicle trajectories as well as a set of synthesized ground truths from routine vehicles. The evaluation results suggest that VeTrac can correctly reconstruct 98% of the trajectories in highway environment and 89% of the trajectories in complex urban environment, which outperforms other baseline methods by at least 32%.

## 1.4 Organization of This Thesis

This thesis has five chapters. In Chapter 2, we first explain related concepts and summarize representative works regarding those concepts. In Chapter 3, we introduce UniLoc, a unified framework that exploits the scheme diversity to gain extra performance improvement from state-of-the-art localization solutions. In Chapter 4, we introduce a last-mile school shuttle planning system that learns the real public transportation us-

age and true travel demands from imperfect crowdsensed trajectories, and then plans need-satisfying routes based on that. Finally in Chapter 5, we introduce VeTrac, a vehicle path reconstruction system that leverage multi-dimensional similarities and graph propagation based modeling to reconstruct individual vehicle paths from low-resolution videos. Finally in Chapter 6, we present the conclusion of this thesis and imagine our future work.

# Chapter 2

# Related Work

## 2.1 Model aggregation

. BMA has been widely used in many applications [51], like health data analysis and weather prediction. It has been proven that BMA can better tolerate the uncertainty in model selection [10]. Locally-weight BMA is studied in classification [37] and applied in the analysis of urban traffic speed from multiple data sources [131]. Unlike the context-based weighting in UniLoc, these works are based on historical data for weight assignment.

## 2.2 Localization error

. CONE estimates the error of one localization scheme by multiple measurements at the same location. However, it cannot be used if the user is moving continuously. GPS accuracy is studied according to the measurement conditions [26]. Multiple regression has been used to study the impact of signal strength values [22]. The error Probability Density Function (PDF) is also studied theoretically [55]. The above works only focus on Wi-Fi RSSI fingerprinting, and do not consider the real-time sensor data. CO-MAP [27, 28] leverages location information of mobile devices to improve their multiple access performance.

A-Loc [69] uses the error models of some localization schemes to select one low-cost scheme that can meet the accuracy requirement. UniLoc is different from a-Loc in two

aspects. First, the error modeling and prediction in a-Loc are not scalable. A-Loc estimates a probability that the user is at one place, and then calculates the localization error of one scheme based on the pre-measured offline error records at all possible locations. As a result, it does not consider temporal variation of environment conditions and cannot be used in new places where no error record has been collected. Second, a-Loc only selects one scheme; whereas UniLoc combines the outputs of multiple schemes to achieve a better result.

## 2.3 Individual localization schemes

. As a unified localization framework, UniLoc is orthogonal to the development of individual localization schemes. Some works [39, 89, 145] reduce the sampling rate of GPS by opportunistically turning on some low-energy-cost localization schemes (e.g., Wi-Fi or cellular RSSI fingerprinting) based on users' routing trajectories.

RADAR [2] is the first Wi-Fi RSSI fingerprinting localization system. Place Lab [16] extends RADAR from offices to metropolitan scale. Horus [128] handles the temporal variation of Wi-Fi signals. In some works, e.g., EZ [17] and EZPerfect [83], the log-distance path loss model is used to estimate user location based on trilateration. Some recent works, e.g., ArrayTrack [124] and SpotFi [60], exploit PHY layer information, like CSI, to provide sub-meter localization. Otsason et al. [88] use cellular RSSIs to perform fingerprinting localization. The cell tower ID sequence is also used to estimate the user's position [90], whereas it can only be used along the users' routing trajectories.

Constandache et al. [20] first exploit the inertial sensors on smartphones to enable PDR in outdoor environments. Li et al. [64] develop a practical PDR system for indoor localization. UnLoc [115] leverages some Wi-Fi and structure signatures as landmarks for indoor PDR. Zee [96] uses Wi-Fi signatures to find the start of trajectories for PDR. LiFS [127] only processes accelerometer data to monitor walk steps, which are further used to construct indoor radio map. FOLLOWME [104] uses magnetometer to identify different indoor pathes which are further used to guide users to the right destinations. Peer-assisted landmarks are exploited to improve the performance of PDR localization in SoundMark [15] and Maxlifd [49].

Tsai [111] incorporates ultrasonic time-of-flight into PDR by Kalman filter. SLAC [50]

and Travi-Navi [141] fuse the Wi-Fi RSSIs and motion-based PDR in particle filters. Travi-Navi also provides some techniques, like image-assisted navigation and path recommendation. The distance constraints between peers are used to adjust Wi-Fi RSSI fingerprinting [70]. Cross-modality training [91] is used for positioning in highly dynamic industrial settings. MapCraft [123] enables reliable indoor map matching for indoor localization and tracking. Geomagnetic field and motion pattern are used for indoor localization [116]. Camera images and inertial sensors are used for localization on smart glass [137].

Guoguo [71] leverages acoustic anchors for fine-grained indoor localization. An energy-efficient localization scheme is developed for wireless sensor networks to monitor patients in a nursing home [97]. WiFi beacons are leveraged to improve the IEEE 802.15.4 localization system [120]. Based on radio frequency (RF) detection, device-free localization systems [125] are developed for residential monitoring. DopEnc [134] measures the relative moving speed between two persons using acoustic signals and further infers whether they interact with each other. strLight [133] uses string lights to deliver information including location information to mobile devices.

## 2.4 Bus stop selection (BSS)

Given the road network consisting of home, school, bus depot and the origin-destination (OD) matrix, BSS seeks to select a set of bus stops and assign students to those stops. According to two comprehensive survey studies [31,92], many works assume that the potential locations of bus stops are given. With that, BSS is then formulated as an assignment problem to minimize the number of bus stops or the total student walking distance. Only a few works solve BSS in conjunction with route planning by heuristics, which can be classified into the following three strategies: the location-allocation-routing strategy [36], the allocation-routing-location strategy [9] and the location-routing-allocation strategy [100]. All above studies did not take into consideration of the existing public transits and assumed that the best pickup locations for students are within walking distance from their homes. This is often not the case in practice given the multiple choices of public transportation and diversity among students home-school trips. Our work learns potential pickup locations for individual students from their daily trips with

public transits, and considers all possible pickup locations with proposed graph-based data structure to ensure a low cost of suggested routes.

## 2.5 Bus route generation (BRG)

Given the selected bus stops and the number of students assigned to them, BRG searches for the optimal routes and is very similar to the vehicle routing problem (VRP) [61]. Due to the problems NP-hardness, only relatively small instances can be precisely solved via optimization algorithms (e.g., dynamic programming, branch-and-bound). Therefore, in practice, classical heuristics that combine a construction heuristic (e.g., the savings algorithm [19], the sweep algorithm [40], and the Fisher and Jaikumar algorithm [40]) and an improvement method (e.g.,$\lambda$ -opt) are often used to obtain a feasible solution. More recently, a significant research effort has been dedicated into metaheuristics such as genetic algorithm [3], simulated annealing [87] and Tabu search [21], which are capable of consistently producing high quality solutions at the expense of speed and simplicity. However, those methods cannot handle the problem considered in our paper where student have multiple potential pickup locations. Directly applying those methods requires constructing and solving a number of VPR instances and results in infeasible computation cost. We thus propose a graph-based data structure to reduce search space and develop a customized Tabu search algorithm upon the graph to construct proper routes.

## 2.6 Common practices in school bus planning

Traditionally, planners have to design school bus routes based on costly surveys and their own expertise [44]. There are two commercial practices regarding school bus planning: door-to-door shuttles and transportation hub expresses. Both methods first require that each student provides a best pickup location (i.e. usually his/her home or a major transportation hub traversed). The bus planning problem is then formulated into an optimization problem such as TSP variants (e.g. mTSP [7]) and VRP variants (e.g. CVPR [61], DARP [21]). Different from above approaches, we learn the best pickup locations for individual students instead of assuming a homogeneous pickup strategy (either homes or transportation hubs) is best for all students. Learning from trajectories can

also reveal the system-wise optima, while surveying students can only obtain knowledge of individual-wise optima.

## 2.7 Data-driven bus route planning

Today, there are several recent projects that leverage information from different data sources to facilitate bus route planning. In [94, 132], researchers learn the metro passengers' final destinations from cellphone data. The bus routes are determined by solving TSP-like optimization problems that are formulated by cluster centers of the learned destinations. In [14, 18], taxi records between two regions are used as an indicator of poor public transit coverage and bus routes are generated to bridge those regions. In [95], similar origin and destination locations of commuters are learnt by a clustering over smart card data, where bus routes are designed to link those locations. In [75], researchers build a transportation mode choice model from both taxi records and bus transactions. With this model, region pairs with low probability of passengers taking buses are identified, where new bus routes are designed by maximizing the expected utilization of public bus service. For those works, they implicitly assume that each passenger has only one origin and destination pair. They cannot be used in our scenarios, because a student usually has multiple potential pickup locations.

## 2.8 Travel mode detection

Most existing methods share a general principle. First, a classification model (e.g Decision Trees, Random Forest, Bayesian Network, Support Vector Machine and neural network) [122] is trained from features of historical trajectories. Then, by feeding mobility features of new traces into trained model, sample-wised travel modes can be determined. Most approaches regard mobility patterns like distance, speed and IMU readings as features, while some recent works involve new feature (e.g. barometer readings [29, 99]) or new information (e.g. GIS information [108]) to improve accuracy. These approaches assume that mobility features possess a consistent error level across samples. However, this assumption is not satisfied in NSE data due to its sparsity and noisiness. Thus directly applying those methods leads to erroneous results.

## 2.9 Path inference

Path inference refers to the problem of inferring precise vehicle trajectories from a sequence of imperfect geo-locations, which are usually obtained from global positioning system (GPS), call detail records (CDR) or geographical check-in data [5]. In practice, common imperfections include low-sampling-rate trajectories, noisy observations and anonymized data.

To reduce uncertainty in low-sampling-rate trajectories, Zheng et al. [138] observed that travel patterns between certain locations are often highly skewed and similar trajectories can often complement each other. With that, they proposed to infer probable routes of a sparse trajectory by leveraging hints from its reference trajectories. Banerjee et al. [5] first learned an edge-weighted graph that captures spatial transition patterns embedded in historical trajectories and then inferred probable trajectories as well as their possibilities by performing random walks on the graph.

To unveil precise locations from noisy observations (also known as the problem of map matching), the key idea is to use information from the whole path instead of one single observation. Many probabilistic algorithms have been proposed with the idea of Kalman filter [30], particle filter [45] and interactive-voting [78]. By further assuming the data following the Markov property, methods that utilize Hidden Markov Model [84] or Conditional Random Field [53] have been exploited.

To reconstruct paths from completely anonymized data (no identify information is associated with geo-locations), a constant speed is usually assumed in [25, 112], based on which different geo-locations are associated to different identities. Instead, Yang et al. [126] proposed five spatio-temporal features - transition randomness, visiting normality, stability, horizon and significance - to group anonymized data points into corresponding trajectories.

## 2.10 Vehicle re-identification (Re-ID)

Inspired by person re-ID, a vehicle re-ID model ranks snapshots in a database (usually taken from different cameras and at different times) by decreasing similarity to a query snapshot. The intent is that any snapshots in the database that are co-identical with the vehicle in the query are ranked highly.

Most vehicle re-ID models rely purely on the appearance cues. This is fundamentally challenging due to 1) the same vehicle observed in different camera views may look quiet different under various camera settings such as resolutions, lighting, poses and viewpoints; and 2) Different vehicles could have very similar colors and shapes, especially for those belonging to one same manufacture. In the meantime, subtle cues for identification, such as license plates and special decorations, are often small in size and sometimes indistinguishable due to the low quality of vehicle snapshots. To overcome above challenges, Liu et al. [73] released the VeRi-776 dataset, which contains over 50,000 snapshots of 776 vehicles captured by 20 cameras under different viewpoints. They proposed a model that integrates visual features including the texture, color and semantic attributes. Wu et al. [121] proposed to alleviate cross-camera variations by calibrating the pre-trained re-ID model for online cameras with positive and negative re-ID pairs that are captured from online videos. Wang et al. [118] emphasized the effectiveness of subtle difference among small regions in vehicle snapshots. They proposed an orientation-invariant feature embedding to align and compare visual features from different snapshots by extracting each snapshot local region features in 20 key points. Zhou et al. [144] employed a viewpoint-aware attention model and an adversarial training architecture to infer multiple-view features from single-view input and improved re-ID task performance by combining original input and generated features.

In recent years, some works exploited the spatio-temporal information to refine the appearance-based re-ID results. Liu et al. [74] re-ranked re-ID results by favoring snapshot pairs that are close to each other in both spatial and temporal domains. Wang et al. [118] employed an additional regularization where the spatio-temporal distance of each snapshot pair is modeled by a logarithmic normal distribution over the transition time between corresponding cameras. Instead of just considering the similarity between each snapshot pair, Shen et al. [102] utilized the most likely path in between to determine their relations. They first generated a visual-spatio-temporal path proposal by optimizing a chain Markov Random Fields model, which is then used as regularization priors for an appearance-based re-ID model.

## 2.11 Multi-camera tracking

Multi-camera tracking aims to determine the position of every vehicle at all times from video streams taken by multiple cameras with non-overlapping views. In order to associate tracks of different vehicles from disjoint camera views, multi-camera tracking usually resorts to spatio-temporal reasoning. Huang et al. [52] first formulated multi-camera tracking into the assignment problem. With the assumption of vehicles traveling in one direction, they presented a probabilistic approach to integrate the colors and sizes of vehicles with velocities, arrival time and lane positions across two cameras. Kettnaker et al. [56] extended this approach to office-like environments with a Bayesian formulation. They required manual input of the topology of a camera network and transition probabilities, which is later relaxed by Makris et al. [80] with a method that learns the topology of a camera network by examining the co-occurrence of entry and exit events across different camera views. More recently, Javed et al. [54] employed kernel density estimators to estimate the probability of a vehicle entering a camera view with a certain time interval given the location and velocity of its exit from another camera view. Matei et al. [81] presented a multi-hypothesis tracker using kinematics, appearance and road network constraints. However, these methods mainly focus on tracking low-density traffic in constrained environments with limit cameras and path options. In the large-scale urban traffic scene, it is difficult to model the spatio-temporal patterns due to the complicated traffic conditions and road networks.

# Chapter 3

# UniLoc: Exploiting the Diversity of Localization Scheme for Improved Individual Mobility Sensing

## 3.1 Motivation

To investigate the performance diversity of existing localization schemes, we run five typical localization programs[1] independently on a smartphone (Google Nexus 5X) along with a daily walking path from our laboratory to a restaurant. Figure 1.2 shows that the path is composed of different segments, including indoors (office, basement passageway, semi-open corridor and car park) and outdoors.

- **GPS**. We use the results reported from the default GPS module on smartphones.
- **Wi-Fi RSSI**. We adopt RADAR [2] for its simplicity and effectiveness. We first build an offline fingerprint database by collecting RSSIs from all audible APs at different locations. We calculate the Euclidean distances between an online measured RSSI vector and all offline fingerprints, and find the location with the shortest RSSI distance.
- **Cellular RSSI**. We use the same fingerprinting algorithm of RADAR on cellular GSM signals like in [88].

- **Motion-based PDR**. Inertial sensors (i.e., accelerometer, gyroscope and magnetometer) on smartphones have recently been exploited for PDR [20, 64]. We implement the system proposed in [64] which infers the walking model (i.e., step count, step length and walking orientation) from the readings of inertial sensors and uses a particle filter to incorporate the map constraints (e.g., path edges and walls). We also detect more landmarks (e.g., turns, doors and Wi-Fi signatures) [115] for calibration.

- **Sensor data fusion**. Some recent solutions perform sensor fusion across Wi-Fi RSSI and motion-based PDR [48, 141]. We adopt the approach in [141] and assign different weights to the particles of motion-based PDR according to the Wi-Fi RSSI distances between the online and offline RSSI vectors.

Figure 1.3 depicts the measured errors of these localization schemes. At every location, each scheme reports an estimated location of the user independently. Since we know the user's true location, the localization error can be calculated. At some locations, the Wi-Fi and cellular based schemes provide identical result, because they use the same fingerprinting algorithm and the same offline fingerprint layout. From the experiment results, we find the following two observations.

*First, none of these schemes can cover the path with stable performance.* Such a conclusion is also partially supported by some indoor experiments [65]. One reason may be that the existing schemes do not explicitly handle the variation of sensor data quality. Even for the fusion-based scheme, Wi-Fi RSSIs cannot always help reducing the localization error of the motion-based PDR. At some locations, e.g., ∼180 m, the low-quality Wi-Fi RSSIs make the estimated location depart from the user's true location. The existing fusion-based schemes [48,141] process the Wi-Fi RSSI data in the same way at different locations, but do not consider the quality variation of Wi-Fi RSSI data.

*Second, different localization schemes complement with each other at different locations.* The performance of each scheme changes spatiotemporally, as the data quality of

[1]The parameters of each scheme are set to be the optimal empirically, e.g., for the motion and fusion based schemes, 3000 particles are generated and maintained every step. In this work, we focus on the localization schemes that are ready to implement on commercial off-the-shelf smartphones. The schemes using other sensors (e.g., bluetooth [1], camera [24] and sound [109]) are not considered, as they require customized hardware devices, such as iBeacon transmitters, programmable LEDs or high-end microphones.

different sensors varies, caused by the natures of physical sensors (e.g., error accumulation of gyroscope) or variation of environment conditions (e.g., variant Wi-Fi AP density or interference [6]). It is hard for a single scheme to overcome the intrinsic limitations of sensor data; however, at each location, it is possible that at least one localization scheme is able to provide good performance. For example, among the total 91 locations of the path, the cellular-based localization scheme provides the highest accuracy at 14 locations (15.4% of the total locations), in which 10 locations (11.4% of the total locations) are in the basement segment, where Wi-Fi and GPS are not available and the error of the motion-based PDR scheme increases accumulatively.

**Design space**. If we can predict the localization error of each scheme at every location, we may choose the most-accurate scheme as our result. Figure 1.3 shows that the performance of Skyline is more stable than any individual schemes. For the Skyline, as we know the true location of the user in experiments, we can calculate the localization error of each scheme; however, in reality, it is hard to predict the localization error online and in turn find the best scheme. Additionally, we want to further ask a question: can we go beyond the Skyline? Can we combine the outputs of all schemes such that the combined result may exceed the Skyline? Note that we do not need very accurate error prediction to achieve the above goals. We do not need the absolute errors of each localization scheme, but just the relative errors that can distinguish their accuracy performance.

## 3.2 Error modeling

The experiments in Section 3.1 have demonstrated that the error of one localization scheme is mainly determined by the online sensor data. For one localization scheme, given a series of measurements, error modeling is to learn the quantitative relation between the localization error and corresponding sensor data. It can be formulated as a regression problem. In this section, we introduce a general error modeling workflow and learn the error models of five typical localization schemes.

### 3.2.1 General workflow

We adopt a general 2-step error modeling workflow, which is applicable for all localization schemes.

*Step 1: Data collection.* We treat all localization schemes as black boxes and execute them on smartphones independently. At every location, we measure some data, including the estimated locations of all underlying schemes and the data from all available sensors. As we know the user's true location, the localization error of each scheme can be calculated. As a result, for each scheme, we build a database that records the localization errors and the corresponding sensor data at different locations.

According to our experiments, also confirmed by many previous works [69], most localization schemes has distinct characteristics in indoor and outdoor environments. To minimize uncertainty in error modeling, we perform the studies in indoor and outdoor environments separately. In this work, we treat all the places with roofs (e.g., corridors on the edges of buildings) as indoor environment, since they have similar error characteristics. IODetector [143] is used to automatically identify the indoor and outdoor environments. It is very energy-efficient, as it only uses some low-power sensors, including light sensor, magnetism sensor and cellular signals.

*Step 2: Regression modeling.* This step is to learn a regression model based on the collected data. We categorize the existing localization schemes into several classes according to the sensor data they use. For each data source, we find some factors that may influence the localization accuracy, according to the experiences learned from some existing works on specific localization schemes. Table 3.1 summarizes the potential influence factors of some typical data sources. The set of influence factors is the same for all the schemes using the same data sources. Different schemes of specific localization algorithms may have different coefficients for the same factor. The fusion-based localization schemes have the influence factors of all involved data sources. We use multiple linear regression model to learn the coefficients. For one scheme, assume we have $N$ entities in the training database. The localization error $y_i$ can be calculated as the follows.

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_p x_{pi} + \epsilon_i, \qquad \text{(Eq. 3.1)}$$

where $x_{1i}$ is the first factor of the $i$th entity, $\beta_1$ is the coefficient of the first factor, $\beta_0$ is the intercept term and $\epsilon_i$ is the residual term after regression. One localization scheme

Table 3.1: Influence factors of typical localization models.

| Models | Schemes | Influence factors |
|---|---|---|
| GPS | GPS on smartphones | Number of visible satellites<br>Geometric positions of visible satellites |
| Wi-Fi RSSI | RADAR [2]<br>Horus [128]<br>EZ [17] | Spatial density of fingerprints ($\beta_1$)<br>RSSI distance deviation ($\beta_2$)<br>Number of audible APs |
| Cellular RSSI | Otsason et al. [88] | Spatial density of fingerprints ($\beta_1$)<br>RSSI distance deviation ($\beta_2$)<br>Number of audible cell towers |
| IMU | Li et al. [64]<br>Travi-Navi [141]<br>UnLoc [115]<br>Constandache et al. [20] | Distance from the last landmark ($\beta_1$)<br>Width of the corridor ($\beta_2$)<br>Orientation changing frequency<br>Step count error |

Table 3.2: Estimated coefficients for four typical localization schemes, i.e., RADAR [2], the cellular-based localization scheme [88], the motion-based localization scheme [64] and Travi-Navi [141].

| Wi-Fi, Cellular, Motion, Fusion | | Estimate | pValue | $\mu_\epsilon$ | $\sigma_\epsilon$ | $R^2$ |
|---|---|---|---|---|---|---|
| **Indoor** | $\beta_1$ | 1.27, 2.08, 0.06, 0.06 | 0, 0, 0, 0 | 0, 0, -0.23, -0.27 | 3.46, 4.37, 0.49, 0.81 | 0.26, 0.42, 0.85, 0.85 |
| | $\beta_2$ | -0.02, -0.16, 0.04, 0.05 | 0, 0, 0.01, 0 | | | |
| | $\beta_3$ | n/a, n/a, n/a, 0.13 | n/a, n/a, n/a, 0 | | | |
| **Outdoor** | $\beta_1$ | 1.01, 2.49, 0.10, 0.10 | 0, 0, 0, 0 | 0, 0, -0.07, -0.07 | 2.57, 15.17, 0.6, 0.6 | 0.31, 0.53, 0.88, 0.88 |
| | $\beta_2$ | -0.12, -0.2, 0.03, 0.03 | 0, 0.09, 0, 0 | | | |

has $p$ influence factors. The residual $\epsilon$ of all entities should follow a normal distribution with a deviation $\sigma_\epsilon$.

### 3.2.2 Error models

We learn the error models for those five localization schemes implemented in Section 3.1. We adopt the procedure suggested in [82] to conduct our regression analysis. We estimate the coefficients and check the model appropriateness in this section. We conduct experiments in an office of 56×20 $m^2$ and an open space of ∼1000 $m^2$ on our campus. In each place, we perform location estimation at 300 locations. Table 3.2 presents the coefficients of error models for four localization schemes, except GPS.

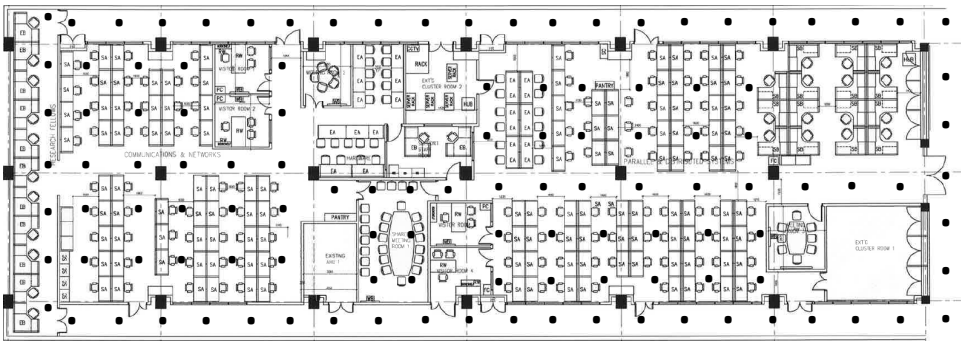


Figure 3.1: Office layout plan (56×20 $m^2$) and offline fingerprints.

**Checking the model appropriateness**. The intercept term $\beta_0$ is zero for all schemes, since the localization error is zero if all coefficients are zero. The results in Table 3.2 suggest that multiple linear regression can approximately capture the variability in localization error. 1) For every localization scheme, we find more than two data features that have a pValue less than 0.05. The pValue evaluates the hypothesis that the coefficient is equal to zero. Normally, a pValue less than 0.05 indicates that the feature is significant given the other features in the model. 2) The residuals of all error models follow a normal distribution with a mean in the vicinity of zero ($\mu_\epsilon$) and a small deviation ($\sigma_\epsilon$). 3) The $R^2$ values of the motion and fusion localization schemes are as high as 85%. It means that the derived model can explain (approximately larger than 85% of) the variability in localization error. Although the $R^2$ values of the Wi-Fi and cellular schemes are low, the experiments in Section 3.4 will show that these error models are sufficient, because UniLoc does not need the absolute errors of each localization scheme, but just the relative errors to distinguish the accuracy of different schemes.

**Wi-Fi and Cellular RSSI fingerprinting**. Spatial density of fingerprints ($\beta_1$) is measured by the average distance between two fingerprints around the location under consideration. The localization error is likely to be high if the fingerprint distance is large. Therefore, the coefficient is a positive number. RSSI distance deviation ($\beta_2$) is the deviation of RSSI distances for the first $k$ location candidates ($k$=3 in our setting) that have the lowest RSSI distances. If the deviation is small, the fingerprints at these locations are more similar, and in turn the estimated location is more likely to be wrong.

In our experiments, the distance between two fingerprints is 1∼3 m. Figure 3.1 presents the layout plan of our office (56×20 $m^2$) and the offline fingerprint locations.

For larger fingerprint distances (e.g., 5 m, 10 m, and 15 m), we downsample the fine-grained fingerprint data. The spatial density is not uniformly-distributed in a large place, due to the physical constraints during fingerprint collection. We use the density around the user's location as the value of factor $\beta_1$. During the training phase, we know the user's true location. For online localization, to calculate the value of factor $\beta_1$, we estimate the user's location based on the existing location prediction methods [76], like Hidden Markov Model (HMM) or Kalman filter. In our current implementation, we use a second order HMM, which can provide an acceptable estimation accuracy.

Our experiment results suggest that the number of audible APs is not a significant factor. When the number of audible APs is less than 3, it is unlikely for the RSSI fingerprinting scheme to provide a meaningful result; as the number increases, however, it does not present a strong correlation with localization accuracy.

Horus [128] handles the temporal variation of Wi-Fi signals by learning a distribution of RSSIs for every audible AP. However, it requires hundreds of samples to capture an accurate distribution at one location. Each path (2.78 km in total) or each place (e.g., shopping mall and office) in our evaluation (Section 3.4) requires tens of days to collect fingerprints with a resolution of $3{\times}3\ m^2$. Like some previous works [65], we assume that a RSSI fingerprint database is updated by service providers or crowdsourcing [96, 127]. In our experiments, each offline fingerprint has one sample from each audible AP. The online localization is made within half an hour after the offline fingerprints are collected.

Besides fingerprinting, Wi-Fi RSSI localization can also use propagation models, e.g., EZ [17] adopts the log-distance path loss model to estimate the distances between multiple users and APs. The distances are further processed to infer the users' location by trilateration. The model-based Wi-Fi localization is not considered in this work, because it only works for multiple users and requires a large number of APs, which may not be practical in some places.

**Motion-based PDR**. The motion-based PDR [64] leverages the map to impose constraints on the user's possible locations. The localization error increases as the distance from the last landmark ($\beta_1$) increases, because the step error accumulates. If a corridor or path is wider ($\beta_2$), it has looser constraint and the localization error is likely to be higher.

Orientation changing frequency does not have significant influence in localization

accuracy. The trembling of the user hand may cause inaccurate orientation inference. However, the random error of orientation readings is averaged to almost zero, as 50 orientation readings are made per second and an average orientation is calculated every 3 s.

Step count error is not a significant factor either. Trembling may cause some jitters in the accelerometer trace, which result in errors of step count inference. We add a compensation mechanism into the localization system [64]. The normal period of one human walking step is from 0.4 s to 0.7 s. If the time duration of one step is less than 0.4 s or larger than 0.7 s, the system will infer a false positive or false negative step, and delete or add one step in the user's trajectory. Our experiments show that such a mechanism can well mitigate the localization error caused by trembling.

When the phone is put in different positions, like on hand or in pockets, it infers different orientations of users. Many existing works [48, 64, 111] handle the measurement offset caused by different phone positions. They normally target at imperceptible tracking. We do not consider the impact of smartphone positions. As a localization system, UniLoc provides real-time positioning service. It is reasonable to assume that users hold their phone on hand for updated location result.

Different persons have different gait patterns, like step frequency and step length. Personalization of step model is considered in [64]. Dynamic time warping is applied during the inference of step count from the accelerometer traces, and the step length are adaptively updated by particle filter. We test with 6 persons, including both females and males with different ages (from 20s to 50s). Benefitting from the personalization of step model [64], the individual difference does not impact the localization accuracy much.

**Fusion-based scheme**. It has all influence factors of the motion-based PDR ($\beta_1$ and $\beta_2$). Spatial density of Wi-Fi RSSI fingerprints ($\beta_3$) is also significant, since fine-grained fingerprints have tighter constraints to the particles of motion-based PDR. RSSI distance deviation becomes insignificant, as the particles of the motion-based PDR may not be located in the grids of the first $k$ location candidates of RADAR.

In outdoor environments, the physical distance between two fingerprints is large (e.g., 10$\sim$20 m), as we may not be able to access some regions (e.g., in the middle of roads or blocked by buildings). The coarse Wi-Fi RSSI information cannot refine the motion-based PDR scheme. Therefore, the fusion-based scheme has the same error model with

the motion-based scheme in the outdoor environments.

**GPS**. The results provided by the GPS module of current smartphones include the user's coordinate, Horizontal Dilution of Precision (HDOP) and the number of visible satellites. HDOP measures the confidence of the reported location, based on the number of visible satellites and their geometric positions. A reliable location estimation requires that the number of visible satellites is larger than 4 and HDOP is less than 6 [69]. Through a series of experiments, we find that the number of visible satellites is $\sim$10.9 and the average HDOP is $\sim$0.9 in the outdoor environment. Moreover, based on our measured data (400 locations in two urban open spaces), the GPS error follows a Gaussian distribution with a mean of 13.5 m and a deviation of 9.4 m. Therefore, the intercept $\beta_0$ is 13.5 and the deviation of residual $\sigma_\epsilon$ is 9.4 for the GPS error model.

**Impact of device heterogeneity**. Two devices may have different RSSI measurements from the same wireless signal, due to hardware heterogeneity. If a person uses a phone that is not the device used for fingerprinting, the localization accuracy may be impacted. There are many works handling the problem of device heterogeneity, like online offset calibration [65, 93]. UniLoc is orthogonal to these algorithms. In our experiments with two smartphones, Google Nexus 5X (Qualcomm QCA6174 802.11ac Wi-Fi 2x2 MIMO Combo SoC) and LG G3 (BroadcomBCM4339 5G Wi-Fi combo chip), we also find an offset between the measured RSSIs. We transfer their RSSI readings of device A and B by an online-learned offset: $RSSI_A = \alpha * RSSI_B + \delta$, where $\alpha$ is close to 1 [93]. We also conduct many experiments for GPS and the motion-based localization. The results show that device heterogeneity does not impact the error modeling in these schemes.

**Modelling Overhead**. Since the relation between localization error and sensor data is only determined by the localization algorithm and does not change according to environment variation, the offline error modeling only needs to be performed once when one localization scheme is first intergraded into UniLoc. The learned error models can be used in new places without retraining. Moreover, multiple schemes can be learned at the same time. The data collection of five localization schemes in two places can be accomplished by one person within one day. According to the experiments in a variety of environments (Section 3.4), 300 measurements are sufficient to learn an acceptable error model that can be used in new places to provide substantial performance gain in UniLoc.

## 3.3 UniLoc

In this section, we present two versions of UniLoc and the techniques to reduce the energy consumption.

### 3.3.1 UniLoc1: selecting the "best" localization scheme

To predict the accuracy of one localization scheme online, besides the absolute error, we also consider the uncertainty in the prediction. When a scheme provides a location estimation at time $t$, its localization error ($y_t$) can be predicted as a variable with Gaussian distribution, $Y_t \sim \mathcal{N}(\mu_t, \sigma_\epsilon)$, where $\mu_t = \beta_0 + \beta_1 x_{1t} + \beta_2 x_{2t} + \cdots + \beta_p x_{pt}$ and $\sigma_\epsilon$ is determined by the residual of the regression model ($\epsilon$). Each $\beta$ is an error model coefficient learnt offline and $x_t$ is calculated online by the real-time sensor data at every location. We estimate the confidence of one localization scheme ($c_t$) in its results at time $t$ as the probability that its localization error is less than a threshold $\tau$.

$$c_t = P(Y_t < \tau) = \int_{x=0}^{\tau} \frac{1}{\sigma_\epsilon \sqrt{2\pi}} e^{\left\{-\frac{x^2}{2\sigma_\epsilon^2}\right\}} dx \qquad \text{(Eq. 3.2)}$$

In our implementation, $\tau$ is set adaptively at different locations, as the average predicted error of all available schemes. For every location estimation, we choose the scheme with the highest confidence as our final result. If a scheme is not available at some locations, it just sets its output to zero and UniLoc will exclude it in calculation temporarily.

### 3.3.2 UniLoc2: locally-weighted BMA-based localization

Let $s_t$ be the sensor readings measured at time $t$, and $l$ be one location. A place is divided into $I$ locations, corresponding to $l_1$ to $l_I$. Assume we have $N$ localization schemes integrated into UniLoc. These schemes are $N$ models, i.e., $M^1$ to $M^N$, in BMA. We calculate the joint probability that the user is at location $l_i$ as:

$$P(l = l_i | s_t) = \sum_{n=1}^{N} P(l = l_i | M^n, s_t) \times P(M^n | s_t), \qquad \text{(Eq. 3.3)}$$

where $P(l = l_i | M^n, s_t)$ is the probability estimated by the $M^n$ localization scheme, and $P(M^n | s_t)$ can be considered as the weight ($w_{n,t}$) of the $M^n$ scheme given the real-time

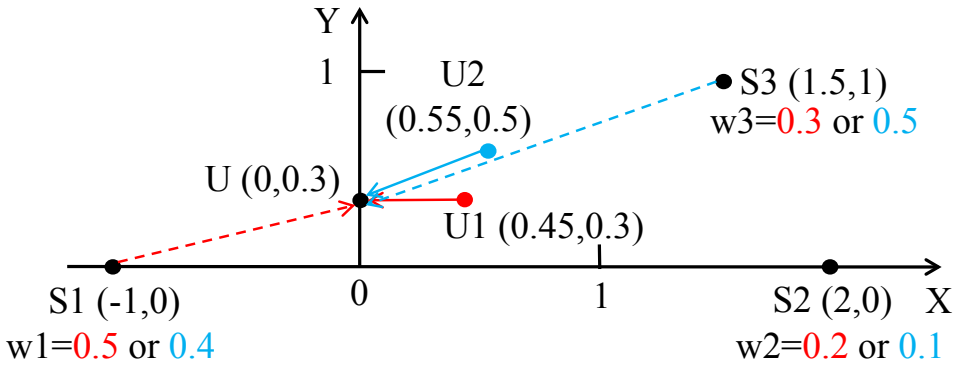



Figure 3.2: The user's true location is at (0, 0.3). Three localization schemes provide three estimated locations (S1, S2 and S3). In two examples, each scheme is assigned with different weights. The weights (w) and localization results (U1 or U2) of two examples are highlighted by different colors (red or blue). The localization errors of UniLoc1 and UniLoc2 are shown with dash and solid lines respectively.

sensor data $s_t$. For a place with $I$ locations, we can estimate the user's location at time $t$ as:

$$L_t = \sum_{i=1}^{I} l_i \times \frac{P(l = l_i|s_t)}{\sum_{i=1}^{I} P(l = l_i|s_t)} \qquad \text{(Eq. 3.4)}$$

There exists an optimal weight for each localization scheme ($w^*_{n,t}$) that minimizes the distance between the user's true location $L^{true}_t$ and the estimated location $L_t$. An optimal weight assignment ensures that one model will have a higher weight, if its estimated result ($P(l = l_i|s_t)$ ) is closer to the true $P^{true}(l = l_i|s_t)$ for all possible locations $l_i$. Therefore, we approximate the optimal weight of one localization scheme ($w^*_{n,t}$) according to its confidence in its result. Among all $N$ localization schemes, the weight assigned to the scheme $M^n$ at time $t$ is calculated as:

$$w_{n,t} = \frac{c_{n,t}}{\sum_{i=1}^{N} c_{i,t}} \qquad \text{(Eq. 3.5)}$$

We use $w_{n,t}$ to approximate $P(M^n|s_t)$ in Equation Eq. 3.3. In a 2D space, we estimate the user's location by calculating her X and Y coordinates independently with Equation Eq. 3.4.

### 3.3.3 Gain of locally-weighted BMA localization

Statistically, BMA can better tolerate the uncertainty in model selection and provide better results than a single model in many applications [10, 51], like health data analysis and weather prediction. In our case, compared with UniLoc1, UniLoc2 has two advantages: *higher localization accuracy* and *better tolerance to the uncertainty in online localization error prediction*. We demonstrate them with two examples, as depicted in Figure 3.2. The first example (in red) assigns right weights to three localization schemes. UniLoc1 chooses S1 as its result, since S1 has the largest weight; however, by combing the results of three schemes, UniLoc2 finds U1, which is much closer to the user's true location. The second example (in blue) has inaccurate weights, as the weights are not inversely proportional to the distances between the estimated locations and the user's true location, e.g., S1 is close to U, but it has a lower weight than S3. Compared with the right weight, an inaccurate weight causes the localization error of UniLoc1 increase significantly, i.e., 0.6% (from 1.04 m to 1.66 m); whereas the error of UniLoc2 does not change much, i.e., 0.31% (from 0.45 m to 0.59 m).

The examples presented in Figure 3.2 are common cases in our experiments. The estimated locations from different localization schemes are scattered around the user's true location. It is rare that all localization schemes provide the results biased to the same direction, since all schemes use different data sets and are executed independently.

Although BMA is widely used in many applications [51] and similar weighted combination is used to fuse the results of multiple Wi-Fi localization algorithms [46], the proposed locally-weighted BMA localization approach differs from the previous works in two aspects. 1) Our approach is locally weighted. Instead of assigning a fixed weight for each scheme globally in a large place [46], we calculate a unique weight for each localization scheme at every location. The spatial environment variation is thus considered. 2) In our approach, the weight of each scheme is determined by the real-time sensor data at every location. The temporary environment variation is also taken into account.

**Discussion**. According to our experiments, compared with UniLoc1, UniLoc2 provides better results. UniLoc1 is a simple solution based on our online localization error prediction proposed in Section 3.3.1. It is mainly used for evaluating the performance of online localization error prediction.

GPS reports the absolute coordinate (i.e., latitude and longitude) in the geographic

coordinate system. Wi-Fi and the motion-based PDR use the local map coordinate. To combine the results of multiple schemes, we convert the result of GPS to the map coordinate by the public digital map information.

### 3.3.4 Energy consumption

The error modeling of UniLoc is conducted offline; thus, it does not consume any energy for online localization. Mobile localization systems consume the energy of smartphones by two operations, i.e., sensor reading and data processing. UniLoc minimizes the energy consumption of both operations.

Most sensors on smartphones, e.g., Wi-Fi, cellular modules and inertial sensors, are energy-efficient [143]. The most energy-consuming sensor is GPS. In UniLoc, GPS is turned off indoors. In outdoor environments, the error of GPS is predicted as a constant (i.e., $\beta_0$, 13.5 m) for all locations based on the error model learnt in Section 3.2.2. The error model does not need any input parameters from the GPS sensor; thus we can predict GPS error without enabling GPS sensor. At every location, UniLoc compares the GPS error with the other schemes. If its predicted error is the smallest one, GPS will be enabled; otherwise, GPS will be disabled.

The computation of UniLoc, including error modeling and Bayesian model averaging, is light-weight, since they only involve simple linear calculation. However, the computational overhead of the motion and fusion based schemes is high, since they need to update the statuses of 3000 particles every 0.5 s. According to our implementation, the updating cannot be accomplished within 0.5 s on Google Nexus 5. We move the particle status updating computation to a server. Sensor data are transmitted to the server via Wi-Fi. If Wi-Fi is not available in some regions, cellular network is used instead, which is pervasively available. The computation of individual schemes and UniLoc is performed on the server.

To avoid data transmissions consuming much time and energy, the raw sensor data are first processed on smartphones. For the motion and fusion based schemes, only small-size intermediate results are transmitted to the server. In our implementation, the high-frequency raw data (50 Hz) from inertial sensors are pre-processed on smartphones to infer the user's step model. The locally-processed results (including moving direction and distance between two updates) are represented by four bytes and transmitted to the

server every 0.5 s. GPS transmits the user's coordinate (latitude and longitude) to the server only if the number of visible satellites is larger than 4 and HDOP is less than 6. Wi-Fi and cellular localization schemes send their online RSSI measurements to the server for fingerprint matching.

## 3.4 Evaluation

We conduct extensive experiments to evaluate the performance of UniLoc.

### 3.4.1 Experiment setting

We aggregate the five localization schemes mentioned in Section 3.1 into the UniLoc framework for testing. All the computation of UniLoc is implemented in C++ running on a workstation with 16GB memory and Intel Xeon E5-1650 v3 processor of six cores. The error models learned in Table 3.2 are used.

We conduct experiments in in different environments, including a campus and urban areas, ∼44047 $m^2$ in total. Most of the testing environments (>89%) are different from the places where the data were collected for training the error models. The urban areas include a floor (95×27 $m^2$) of a shopping mall and an urban open space (∼5700 $m^2$). In these two places, 10 different 30-m trajectories are studied for the motion and fusion based localization schemes; at the same time, the other localization schemes is performed every 3 m along the trajectories. Each place has thus ∼100 tested locations.

### 3.4.2 Error model validation

Based on the derived error models, we predict the localization error of each scheme at one location as:

$$\hat{e}_i = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p, \qquad \text{(Eq. 3.6)}$$

where the coefficients $(\beta_0, \beta_1, \cdots, \beta_p)$ are the results in Table 3.2 and the value of each factor $(x_1, x_2, \cdots, x_p)$ are calculated by the real-time sensor data. If we have $M$ tuples of localization error and sensor data to perform validation ($M$=200 for each test), the normalized Root-Mean-Square Error (RMSE) of the predicted localization error is:

Table 3.3: Normalized RMSE of the online error prediction for different localization schemes.

| Prediction accuracy | Same places | | New places | |
|---|---|---|---|---|
| | Same devices | Different devices | Same devices | Different devices |
| GPS | 0.58 | 0.60 | 0.60 | 0.58 |
| Wi-Fi | 0.65 | 0.84 | 0.84 | 1.17 |
| Cellular | 0.64 | 1.06 | 1.12 | 1.17 |
| Motion | 0.20 | 0.22 | 0.28 | 0.35 |
| Fusion | 0.39 | 0.42 | 0.44 | 0.53 |
| Average | 0.49 | 0.63 | 0.66 | 0.76 |

$$RMSE = \frac{\sqrt{\frac{\sum_{i=1}^{M}(\hat{e}_i - e_i)^2}{M}}}{\bar{e}}, \qquad \text{(Eq. 3.7)}$$

where $e_i$ is the groundtruth of localization error for the $i$th measurement, and $\bar{e}$ is the average localization error of all measurements in the test database.

Table 3.3 presents the normalized RMSE of the predicted error for the five localization schemes. On average, the prediction RMSE is less than 49%, if we use the error model derived by the same device and in the same place. We also collect the validation data with another smartphone model (LG G3) and in some new places where the error model is not trained (shopping mall and another office for the indoor test, and an urban open space). With the new device in a new place, the average prediction RMSE increases to 76%. Although the error models cannot provide perfect prediction of localization error, the results in Table 3.2 are useful in our framework. The error models learned by our approach can be used in new unknown places without re-training. Our experiments in Section 3.4.3 will show that even with imperfect error prediction, UniLoc is able to achieve significant performance gain.

### 3.4.3 Accuracy

We use UniLoc to provide real-time positioning service over eight paths, which are some daily paths taken by the students and staffs on our campus, e.g., from an office to a library, restaurants, bus stations, or an auditorium. Figure 3.3 illustrates the eight paths

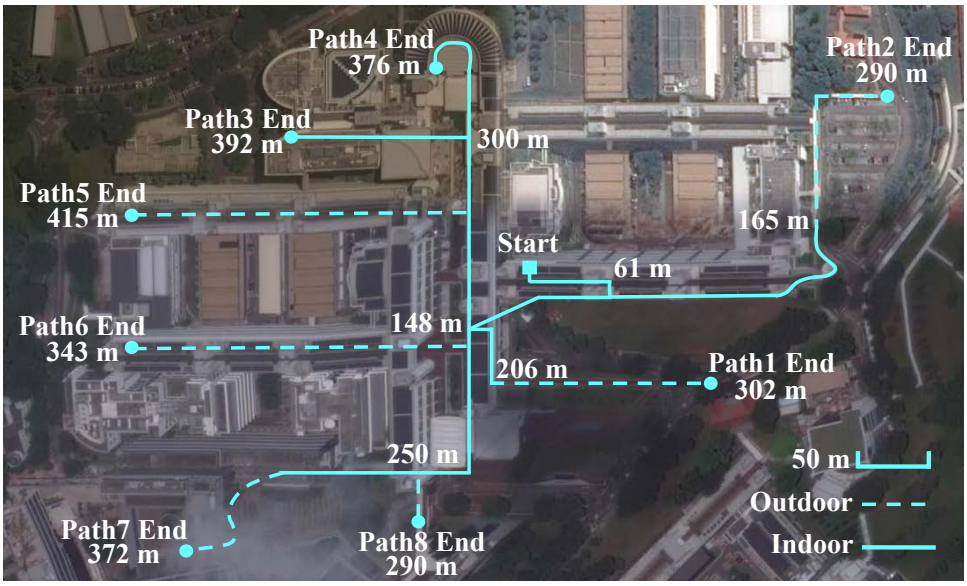



Figure 3.3: Eight daily paths on our campus.

which have a total length of 2.78 km, including 0.8-km outdoor segments and 1.98-km indoor segments. The path studied in Section 3.1 is Path 1 in Figure 3.3. For Wi-Fi and cellular based schemes, the distance between two fingerprints is 1∼3 m in the indoor environments and 10∼20 m in open spaces respectively.

### Overall performance on eight daily paths

Figure 3.4 presents the Cumulative Distribution Function (CDF) of the localization errors for all schemes along the eight investigated paths. UniLoc1 substantially outperforms all the individual schemes, including the fusion-based localization scheme. Although UniLoc1 cannot find the best scheme at some locations due to imperfect online error prediction, UniLoc2 can better tolerate the uncertainty in online error prediction, and achieve comparable performance with the Skyline.

For the 50th percentile value of the localization error, the fusion-based localization scheme provides the smallest error among all the schemes. UniLoc1 reduces the error of the fusion-based scheme by 1.4× and UniLoc2 further improves the reduction factor up to 1.6×.

The 90th percentile value of the localization error from RADAR (i.e., 10.6 m) is much smaller than the motion and fusion based schemes (i.e., 15.3 m), as the latter's

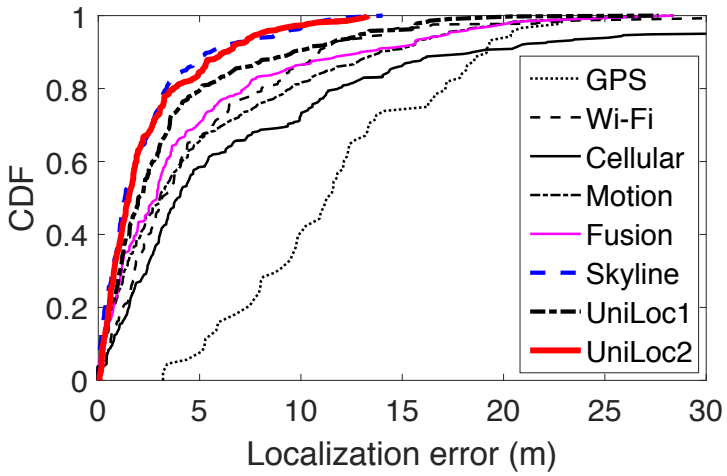



Figure 3.4: Localization error on the eight daily paths.

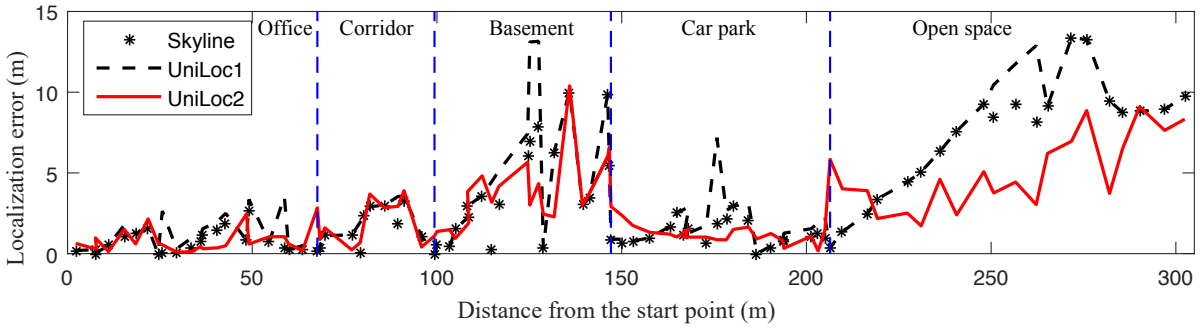
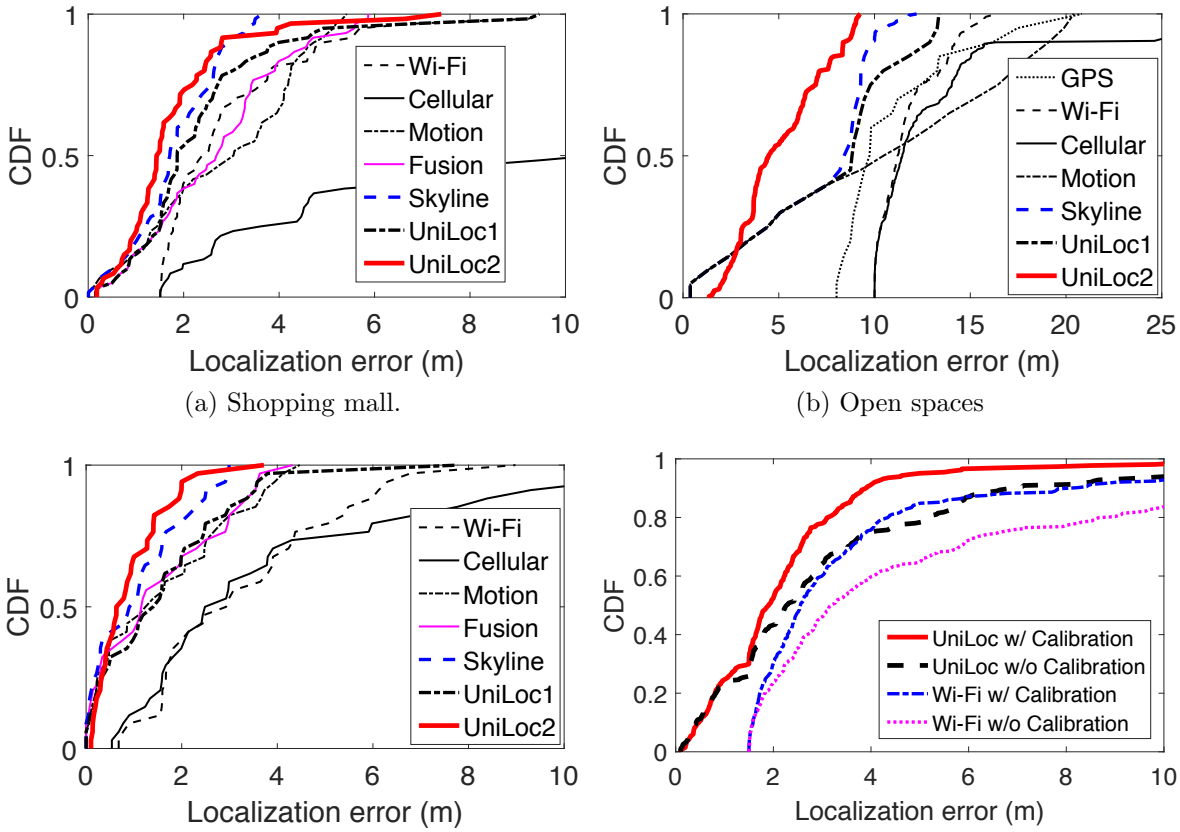



Figure 3.5: Localization error of the optimal single-selection solution and UniLoc along the daily path 1.

error increase if no calibration signatures can be found, e.g., the long straight path of the outdoor segment in Path 1. Even though the Wi-Fi signatures proposed in [115] are implemented, it is hard to find sufficient signatures outdoors. By combining the results from more localization schemes, like GPS and cellular, UniLoc2 controls the 90th percentile value of the localization error as low as 5.8 m, which is 1.8× lower than RADAR.

### The daily path

We revisit the daily path studied in Section 3.1 to analyze the gain of UniLoc in details. Figure 3.5 depicts that UniLoc1 can find the best localization scheme and UniLoc2 outperforms the Skyline at many locations, especially in the outdoor environments, where




(a) Shopping mall.

(b) Open spaces

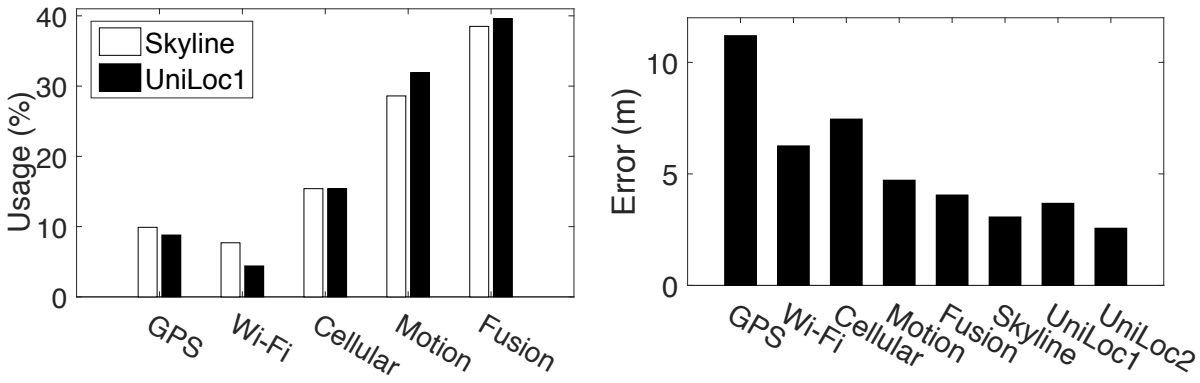
(c) Office.

(d) Heterogeneous devices.

Figure 3.6: Localization error for all underlying schemes and UniLoc in different places or with heterogeneous devices.

the localization errors of individual schemes are large. The optimal single-selection solution, noted as "Skyline", is assumed to know the true localization error of each scheme. Given the result of the best scheme, the other schemes can help moving the combined result closer to the true location.

Figure 3.7a shows that the usage of different localization schemes in UniLoc1 is close to the Skyline. Even with imperfect online error prediction, UniLoc1 can make the right selection, as long as the predicted error can distinguish the accuracy of underlying schemes. In addition, even though UniLoc1 makes suboptimal decision sometimes, the performance of the best two or three schemes are close to each other in these cases, and the misclassification between them will not impact the localization accuracy of UniLoc1 much. Along the path, the usage of the Wi-Fi based scheme is low, because the fusion-




(a) Usage of different localization schemes. (b) Average localization error

Figure 3.7: Evaluations on the daily path 1.

based scheme is selected instead when the Wi-Fi sensor data quality is high, especially in the indoor environment.

Figure 3.7b presents the average error of all localization schemes along the path. Among these schemes, the fusion-based scheme [64] provides the lowest localization error, i.e., 4.0 m, and UniLoc1 achieves a better accuracy, i.e., 3.7 m. By combining the results of all underlying schemes, UniLoc2 achieves an average localization error as low as 2.6 m. It reduces the localization error of the fusion-based localization scheme by 1.7× and outperforms the Skyline by 1.2×.

**Different environments**

Figure 3.6a-3.6c show the CDF of the localization errors for all underlying schemes in the shopping mall, the urban open space and our office respectively. In all three places, compared with the individual schemes, UniLoc2 provides significant performance gain, i.e., ∼1.7× for both the 50th and 90th percentile values of localization error, as it can benefit from all underlying localization schemes. Although the error models are learned in the office and our campus, UniLoc can provide comparable performance in the crowded shopping mall and the urban open space.

Comparing the performance among these places, we find two observations. 1) All these systems have better performance in office rather than shopping mall, as the office has more stable wireless signals and narrow corridors with many turns. The localization accuracy of the cellular-based scheme is low in the shopping mall, because it is at the

basement floor and we can only receive the signals from two cell towers on average. 2) In the outdoor environment, the localization errors of all existing schemes are high and unstable, due to low spatial density of fingerprints or wider paths.

#### Heterogeneous devices

All the above experiments are conducted with a same smartphone model, Google Nexus 5X. We further evaluate the performance of UniLoc with heterogeneous devices. We conduct online localization experiments with another phone, LG G3. The setting of UniLoc is not modified. The error models are still the ones in Table 3.2, which are learned with the data collected by Google Nexus 5X. For the Wi-Fi based scheme used in UniLoc, the offline fingerprint database is also collected with Google Nexus 5X.

Figure 3.6d shows the localization errors of UniLoc and RADAR. Benefitting from the online offset calibration, both UniLoc and RADAR significantly reduce the localization errors caused by the new device, especially when the error is large (1.9× for the 90th percentile value of the localization error). The reduction factor of UniLoc is comparable with the performance gain of RADAR. It means that UniLoc can assimilate the gain produced by the device heterogeneity handling algorithm of individual localization schemes.

### 3.4.4 Energy consumption

We use a Monsoon power monitor to measure the power consumed on smartphones. As the battery of Google Nexus 5X cannot be opened to connect with the power monitor, we use Samsung Galaxy S2 i9100 for power measurement. Changing the phone model does not alter the relative energy consumption of UniLoc and the underlying localization schemes.

The power and energy consumption of every localization system over the daily path 1 are presented in Table 3.4. Along with the 302-m path including an outdoor segment of 96 m, the average energy consumption of the individual localization schemes (except GPS as it is turned off indoors) is 172.5 J, and UniLoc consumes slightly higher energy, i.e., 191.2 J. The most energy-efficient localization scheme is the motion-based PDR. Compared with it, UniLoc only increases the energy consumption by 14%. The data transmissions with Wi-Fi or cellular network do not increase the energy consumption, as the transmission time is short.

Table 3.4: Power and energy consumption of UniLoc and all localization schemes along the daily path 1.

| Schemes | | Localization (mW) | Time (s) | Energy (J) | |
|---|---|---|---|---|---|
| GPS | | 613.7 | 128 | 78.6 | |
| Wi-Fi | | 433.9 | 403 | 174.9 | |
| Cellular | | 415.0 | 403 | 167.3 | |
| Motion | | 418.2 | 403 | 168.6 | |
| Fusion | | 444.2 | 403 | 179.0 | |
| UniLoc | w/o GPS | 444.2 | 354 | 157.2 | 191.2 |
| | w/ GPS | 691.8 | 49 | 33.9 | |

Table 3.5: Average response time for one location estimation, including computation and transmission.

| Schemes | Time(ms) | | | UniLoc | Time(ms) |
|---|---|---|---|---|---|
| | Localization | | Error prediction | | |
| | Phone | Server | | | |
| GPS | 1.1 | 0 | 0.1 | Upload | 72 |
| Wi-Fi | 8.5 | 25 | 6.0 | Computation | 50.6 |
| Cellular | 8.6 | 18 | 6.0 | BMA | 0.1 |
| Motion | 27.5 | 14 | 0.1 | Download | 63 |
| Fusion | 27.5 | 23 | 0.1 | Total | 185.7 |

The current consumption of the fusion-based scheme and UniLoc is the same. In the experiment, we assume UniLoc is used in normal cases where cellular is always enabled to mimic the normal usage of a phone as a user. The extra energy consumption of UniLoc mainly comes from GPS. UniLoc successfully minimizes the usage of GPS, i.e., turning off GPS when its error is expected to be large. In the outdoor environment, compared with the default GPS scheme, UniLoc reduce the energy consumption by 2.1×.

### 3.4.5 Response time

Table 3.5 shows the decomposed response time for one location estimation, including computation and data transmission. The response time mainly includes data transmissions and the computation on both the smartphone (sensor reading and pre-processing) and the server (execution of localization algorithms, error prediction of each localization algorithm and BMA). Since the algorithms of all underlying localization schemes are executed on the server in parallel, the computation time of UniLoc is the time taken by the slowest localization scheme, i.e., 50.6 ms from the fusion-based scheme.

UniLoc only needs 185.7 ms to estimate the user's location once, from the beginning of data sensing to the displaying of location result on the user's phone. It can provide real-time positioning service on smartphones. The computation added by UniLoc is only 6.1 ms, including 0.1 ms for BMA and 6.0 ms for error prediction. The data transmissions of UniLoc occupy 73% of the total response time.

# Chapter 4

# Last-mile School Shuttle Planning with Crowdsensed Student Trajectories

## 4.1 Motivation

We introduce the crowdsensing platform used in this study and demonstrate opportunities as well as challenges to utilize the crowdsensed data for last-mile school bus planning.

### 4.1.1 The National Science Experiment (NSE)

NSE [85] is a nation-wide experiment of Singapore that mobilizes the government and social forces to experiment a large scale mobile crowdsensing system. A special designed mobile device is developed and assigned to a student during school days [101]. The device is equipped with a variety of sensors to measure the motion and environmental parameters, including three dimensional accelerations, light, temperature, noise levels, air pressure, etc. The data are sensed periodically and uploaded to the server opportunistically whenever the device connects to the wireless@SG WiFi hotspots (15,000+ free hotspots covering major public areas in Singapore [119], sponsored by SingTel [106] for free). In addition, the device also scans and sends back the signal strengths (RSSIs) from nearby WiFi hotspots and a third-party localization service from Skyhook [107] is invoked to determine the geolocations from its geo-WiFi database. The average time

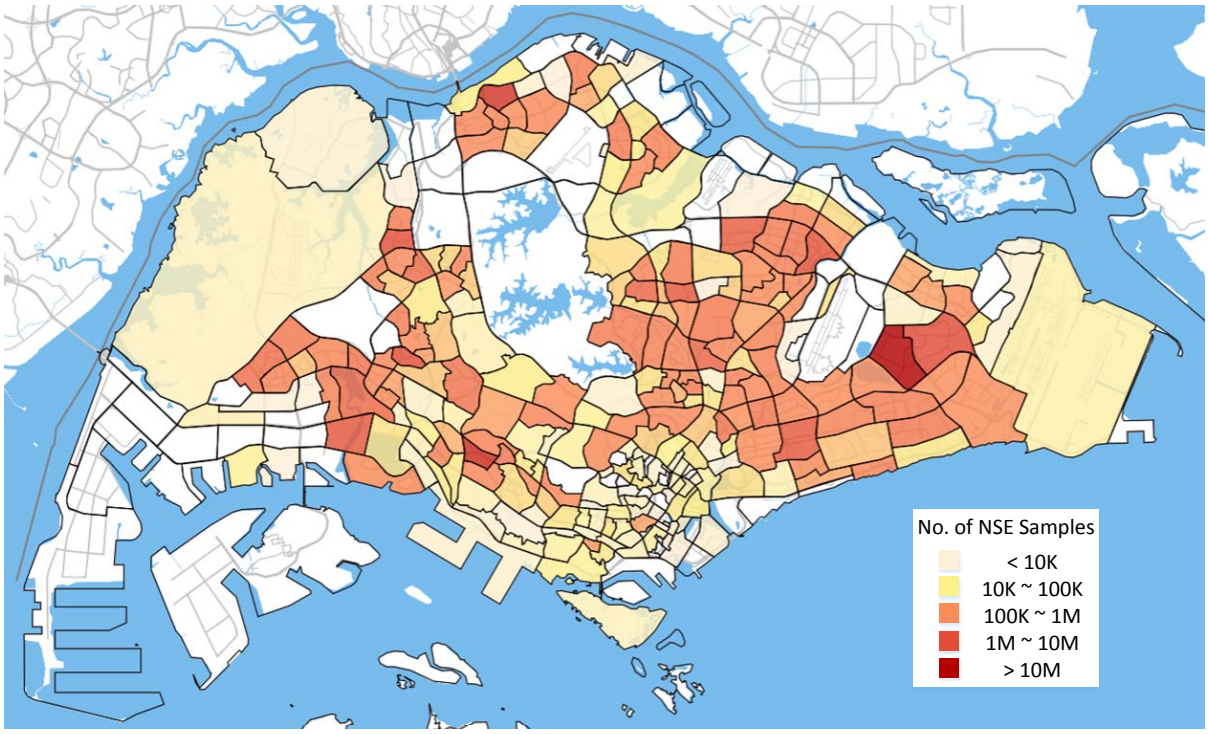



Figure 4.1: NSE Trajectories Spatial Distribution

interval between geolocation updates is 15 seconds. Figure 4.1 visualizes the spatial distribution of all NSE samples (188,100,399 samples in total in one semester).

In this study, we primarily make use of the geolocations of each device which constitute a mobility trajectory of a specific student. The trajectory data have the following two distinctive advantages compared with traditional survey based investigation or origin-destination based data profiling:

- **Data representativeness.** NSE trajectories consists of independently selected students from each participating school. The data group is representative to plan the last-mile school shuttles for each school.

- **Data richness.** NSE trajectories offer detailed trip profiles of individuals - the intermediate location updates during the trip as well as the time taken between those locations. With detailed understanding of transport choices of individuals, we are able to cross study the last-mile shuttle planning together with existing public transportation alternatives.

### 4.1.2 Challenges

Previous works [14, 18, 75, 132] are subject to inaccurate approximation on how and where the students may take a school shuttle. The crowdsensed NSE trajectory data brings opportunities to extract all potential pickup locations for each student. Such consideration factors in the students' route preferences and choices of available public transits, so the corresponding shuttle planning is based on user preference and at the same time offers higher freedom of optimization. Nevertheless, special challenges need to be carefully addressed to develop an effective and efficient solution.

*Challenge 1: Trajectory profiling from imperfect trajectories.* To extract potential pickup locations for each student, we need a detailed profile including precise traveling path and transportation modes on different path segments. However, NSE data are limited in localization accuracy. The geolocations of NSE data are estimated by WiFi hotspot based localization, which is not released by the third-party company [107]. The localization error is inevitable and often higher than those of GPS based approaches. Figure 4.2b shows how the derived locations (denoted as white dots) deviate from the real traveling path (denoted in red lines). According to our assessment, the NSE localization error ranges from a few meters to hundreds of meters with an average of 120 meters. At the same time, the locations are not evenly updated because there are inadequate number of audible WiFi hotspots in certain areas. As a result, the trajectory data have uneven location granularity, as Figure 4.2c suggests.

It has been known difficult to accurately map a sequence of coarse locations to a trajectory on the road map [84]. It is also difficult to accurately detect the transport mode with a trajectory of low resolution location samples [122]. In this paper, we develop a novel approach to generate a representative travel profile for each student. It combines the NSE trajectory data and public transits information via Google Directions Service [43].

*Challenge 2: Embedding all pickup locations in an efficient data structure.* Previous works implicitly assume one specific pickup location for each student, which cannot be extended to handle the problem that each student having multiple potential pickup locations. Blindly applying existing algorithms would result in unacceptable computational cost, e.g., a problem with 500 students and each having 20 potential pickup points incurs the steps of selecting one possible pickup location for each student, and then running the algorithms once for each one out of $20^{500}$ possible combinations. In this paper, we

(a) A sample trajectory (b) Localization drift (c) Sparsity

Figure 4.2: Imperfection of NSE data

propose a novel graph-based data structure that embeds all potential pickup locations of different students into the road networks. Similar transport demands of different students can thus be aggregated, and representative pickup locations can be derived to facilitate bus route optimization.

*Challenge 3: Computationally feasible bus route planning.* Even with the aggregated transport demands, brute force searching for the best bus route is still infeasible, which we prove being NP-hard (in Section 4.2.3). We extend the idea of Tabu search algorithm [41] - a metaheuristic originally designed for guiding a search to overcome local optimality in combinatorial optimization problems. The effectiveness of a tabu-based algorithm requires an application-specific design of its core components, however there is no graph-related design of tabu components. With the observation of convergent mobility pattern of students, we propose a tabu-based expansion algorithm which defines tailored tabu components under the graph structure and can efficient yield close to optimal bus routes.

## 4.2 Design

We design a holistic system for last-mile school shuttle planning to tackle the above three challenges. Figure 4.3 depicts the architecture of proposed system which consists of three key components, i.e., trajectory profiling, graph construction and graph-based route planning. First, trajectory profiling learns real transportation usage and extract potential pickup locations for every student, which offers an extra dimension of optimization and

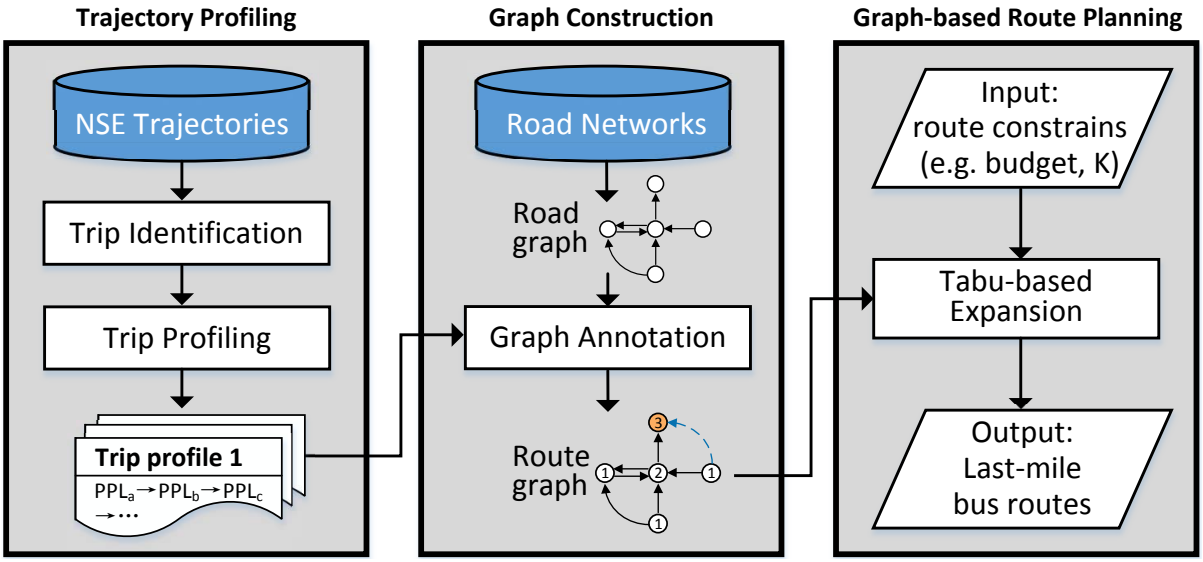



Figure 4.3: System overview

more need-satisfying bus routes can thus be derived. To handle extremely large search space brought by massive potential pickup locations, graph construction constructs a graph-based data structure that aggregates similar demands with the awareness of road networks; Finally, graph-based routing planning efficiently plans the routes that are able to balance the commute time saved for all students and the costs of the shuttles.

### 4.2.1 Trajectory Profiling

Potential pickup locations of a student include the home, the entries and exits of used public transits (e.g., bus stops, metro stations) and all the road segments walked. To extract those from NSE trajectories, we need to infer the precise traveling paths and transportation modes on different path segments.

Conventional methods solve the above problem by combining travel mode detection [122] and map matching [78,84]. To infer the sample-wise travel modes, a classification model is normally trained from labeled historical data in terms of mobility features (e.g., distance, speed and acceleration). Many classification models can be used, such as Decision Trees, Random Forest, Bayesian Network, Support Vector Machine and Multilayer Perceptron. By feeding a target trajectory into the trained model, samples lying in the interchange of different travel modes are selected. Finally, map matching replaces the identified samples with corresponding points projected to the closest road segments.

However, according to our study, the above process performs poorly due to the two characteristics of NSE data (as suggested in Figure 4.2). First, most travel mode detection algorithms require consistent accuracy of inferred mobility features [122]. However, NSE data only provide coarse-grained and inconsistent mobility information. For example, due to the large localization error, the distance between two consecutive samples can be as large as several hundred meters, which results in overestimated speed. Large localization drifts also lead to inaccurate approximations for real locations. Second, due to the data sparsity problem, we cannot exploit potential pickup locations in some areas without enough samples.

We propose a two-step algorithm for trajectory profiling: 1) trip identification first detects the origin (home) and destination (school) of each student from NSE data; 2) trip profiling then infers precise traveling paths, transportation modes and thus potential pickup locations by combining the NSE trajectories and public transits information via Google Direction Service [43].

*1) Trip identification.* A trip refers to a commute trajectory between home and school. Raw NSE trajectories are highly skewed: a small number of samples (2.1% in NSE dataset) contains useful trip information, but the vast majority of samples are "stay" points, where the students stay for a long time (like at homes or schools). These "stay" points provide little travel information, but lead to high computational overhead and false positives when extracting potential pickup locations. Trip identification clusters nearby "stay" points as one representative point so that valid trips are consisted of informative points.

Conventional algorithms normally detect "stay" points by clustering all points with either distance [142] smaller or density [32] larger than a threshold as one point. They do not perform well in processing NSE trajectories, due to 1) SENSg sensors use the received signal strengths from city-wide Wi-Fi access points to determine the device locations. The localization error is large and varies dynamically in space and time. It is hard to decide a global distance threshold as random noises could result in many false-positive identifications. 2) SENSg sensors automatically enter the sleep mode after 15-minute inactivity of movement. Some stay points may not have enough samples to be clustered by the density threshold even though the device did remain there for a long time.

Table 4.1: Success rate of real route extraction by Google Directions Service. The success rate of the Google path set indicates the probability that the set contains the real path traversed by a student. The success rate of the heuristic selection methods indicates the likelihood that the method outputs the real path.

| Trajectory type | | Walking | Public Transits | Driving | Overall |
|---|---|---|---|---|---|
| Google route set | | 91% | 100% | 89% | 93% |
| Heuristics | Distance | 0% | 0% | 44% | 15% |
| | Duration | 25% | 0% | 33% | 19% |
| | Walking | 69% | 50% | 0% | 40% |
| | Transfers | 69% | 25% | 0% | 31% |

Therefore, we adopt a time-weighted density clustering algorithm. Different from conventional algorithms, we use both density and time duration as the metric to identify "stay" points. Specifically, points for each student are classified as core points, reachable points or outliers based on the following criteria: 1) A point $p$ is a core point if the total duration of nearby points (within 200 meters) exceeds an hour. Those nearby points are said to be directly reachable from point $p$. 2) A point $q$ is reachable from $p$ if there is a path $p1, ..., pn$ with $p1 = p$ and $pn = q$, where each $pi + 1$ is directly reachable from $pi$. 3) All points that are not reachable from any other points are outliers. Stay point clusters are formed by core points and all points that are reachable from them. We use the centroid locations to represent each cluster and identify valid trips.

*2) Trip Profiling.* With the home and school locations obtained from trip identification, we use the Google Directions Service [43] to generate all possible paths for each home-school pair. The service returns several well-segmented paths. Each path contains travel mode (i.e. walking, driving and taking public transits), precise intermediate locations and the estimated trip distance and duration. For each home-school pair, 9 paths can be suggested in general. The set of all suggested paths contains almost all reasonable choices, and usually includes the real path traversed by students. We manually label the real paths of 20 randomly selected students and their travel modes along the paths. The results in Table 4.1 indicate that 93% Google sets contain the real path and it works perfectly for students taking public transportation.

Next, we find the real path from the Google set. Google recommends the best route by 4 different heuristics, i.e., the route with the shortest distance, minimum duration, the

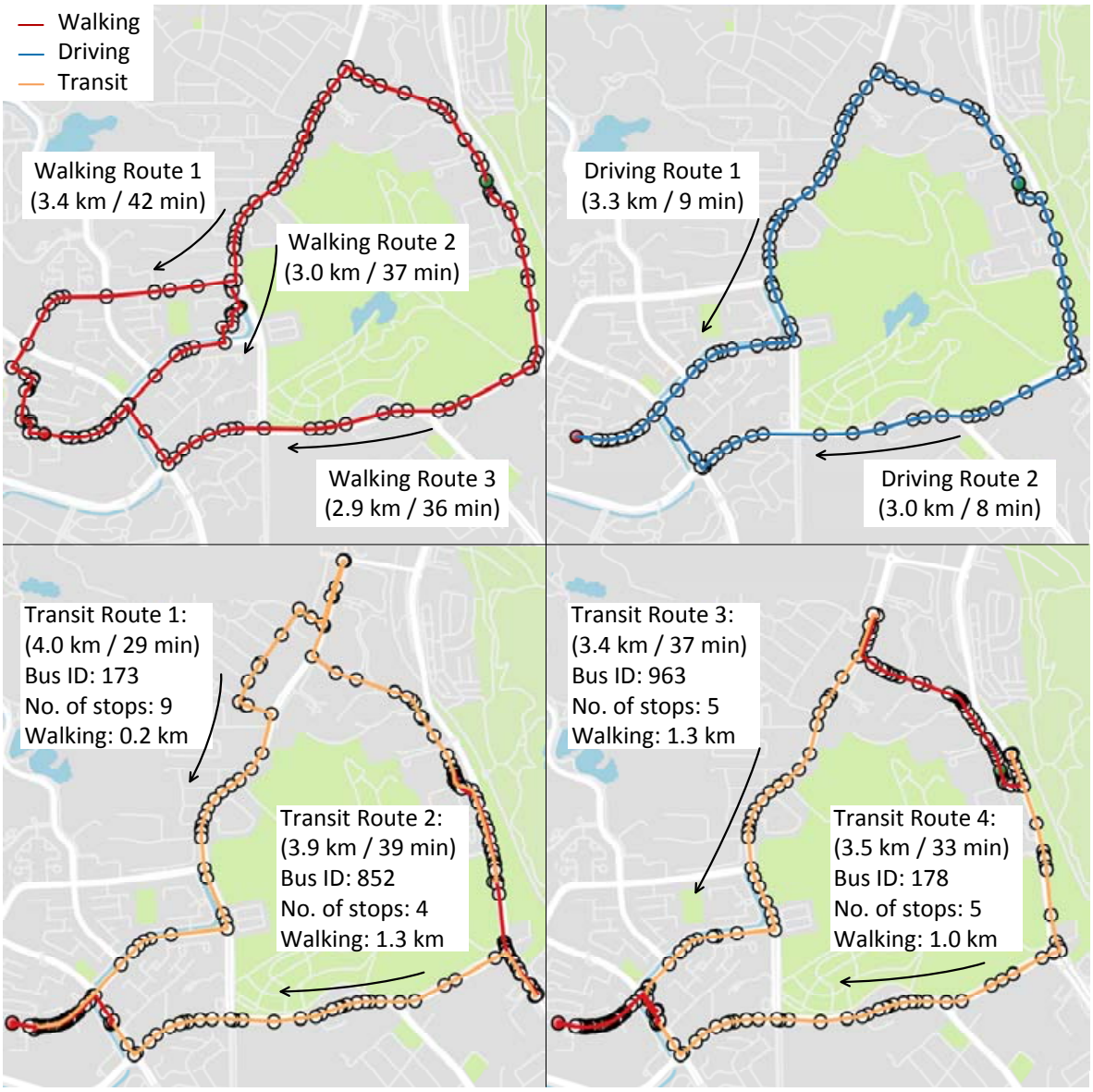
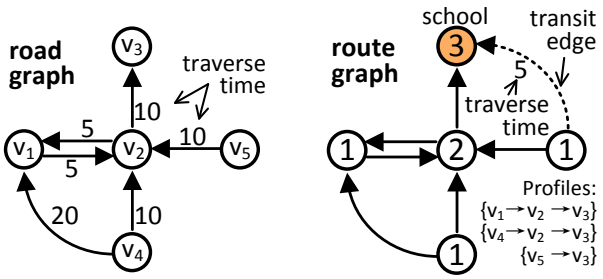



Figure 4.4: Routes suggested by Google Directions Service.

shortest walking distance, or few transit transfers. As shown in Table 4.1, none of these heuristics can match the real routes traversed by students with a reasonable accuracy.

Therefore, for each NSE trajectory, we find its most similar path from the Google set by a hierarchical rule-based classifier with following features: 1) Path shape. We implement a fast approximation Dynamic Time Warping (DTW) algorithm [98] to measure the similarity between two paths. 2) Time duration. It is for situations when Google paths with different travel modes have similar shapes. 3) Distance. For short trips, the




(a) Build the road graph (b) Annotate profiles

Figure 4.5: The construction of the route graph

driving and walking Google paths tend to have similar shape and commute time. In that case, the walking path is of higher probability.

In this way, we infer the precise paths that are well segmented by transportation modes and thus extract all potential pickup locations for each student.

### 4.2.2 Graph Construction

We describe three key properties of effective last-mile bus design and how previous works fail to address them.

- **The capability of structuring multiple potential pickup locations for each student.** Previous works oversimplify students' demands and model each student as a single point of VRP. When each student has multiple pickup locations, such structure cannot select the best pickup location for each student without calculating best routes for all possible combinations (500 students each having 20 potential pickup points will result in $20^{500}$ combinations). Worse, changes in even one student's demands require recalculating all those combinations. Thus, a good data structure should be able to simultaneously represent all potential pickup locations, aggregate similar ones and adapt to small changes.

- **The awareness of road networks.** Previous works employ simple proximity models such as Euclidean-based or grid-based model. Those models incur inaccurate distance estimations as geographically proximate locations could be far by walk due to road constraints(e.g. highways, one-way streets). The awareness of road networks imposes those constraints on shuttle bus design.

Table 4.2: Skeleton of the route graph

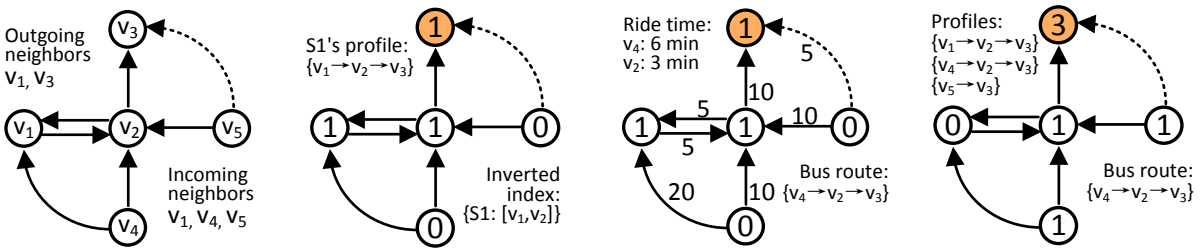
| Structure | Attributes | Description |
|---|---|---|
| Vertex | id | Vertex ID |
| | type | Either "road" or "school" |
| | student_set | Associated students |
| | shortest_time | Shortest time to the school |
| Edge | id | Edge ID |
| | type | Either "road" or "transit" |
| | from_v | Origin vertex ID |
| | to_v | Destination vertex ID |
| | traverse_time | Time needed to traverse through |
| Data | inverted_index | e.g. {Student 1: [$v_1$, $v_2$, $v_3$]} |
| | tabu_list | tabu vertices |

- **The awareness of existing transits.** Previous works aim at standalone services. The ignorance of existing public transits results in high-cost or replicated services. The awareness of existing transits ensures practical shuttle bus design.

To address above problems, we proposed a new graph-based data structure named the route graph, which can be built via the following three stages.

*Stage 1: generate a road graph from the road networks.* As shown in Figure 4.5a, a road graph $G_{road} = (V, E)$ is a directed graph built from road networks, where a vertex set $V$ represents all road segments and an edge set $E$ denotes the linkage and physical direction between road segments. We first extract information about each road segment (i.e., locations of origin and destination, name, length, category, accessibility for vehicles) from OpenStreetMap [86], which is later used to derive the linkage and walking distance between adjacent road segments. We store these information as the attributes of Structure Vertex and Edge (summarized in Table 4.2). For each road segment $v_i \in V$, attribute "student_set" stores students that can get on our bus at $v_i$ and is initialized as an empty list. For each edge $e_j \in E$, we set the edge type as "road" indicating physical connectivity in road networks and the "traverse_time" is estimated by the walking time from one road segment to the other. In this way, the data structure is aware of road networks.

*Stage 2: build a route graph with graph annotation.* A route graph $G_{route} = (V, E, D)$




(a) Neighbors of $v_2$ (b) Insert the profile of S1 (c) Estimate the ride time (d) Allocate a graph

Figure 4.6: Examples of supported operators in proposed data structure $G_{route}$

is built by annotating a road graph $G_{road}$ with students' demands, where three types of information are embedded - the school vertex, potential pickup locations and the usage of public transits. The school vertex is learnt from all input profiles and annotated by setting the vertex's attribute *type* to "School". Potential pickup locations of each student are sequentially annotated to corresponding vertices. For each pickup location, the student ID will be added to the attribute "student_set" of the corresponding vertex. During this process, if consecutive pickup locations are not connected by a direct edge, the usage of existing transits is identified (as shown in Figure 4.5b). A new edge will be added to $G_{road}$, where the *type* is "transit" and *traverse_time* is estimated from the duration between corresponding pickup locations. To facilitate an efficient search of a student on $G_{route}$, we build an inverted index that maps students to associated vertices. The inverted index is a dictionary organized by student ID and stored in class Data as shown in Table 4.2. Figure 4.5b shows the annotation result of three student profiles.

*Stage 3: estimate the time to the school vertex.* After graph annotation, $G_{route}$ is now aware of both the road networks and the existing transits. We thus estimate the shortest time from each vertex to the school vertex by Dijkstra algorithm, which is then stored in vertex attribute *shortest_time*.

Figure 4.7 visualizes the route graph we built for the entire city of Singapore, which consists of 384653 vertices and 597446 edges. The route graph sketches the contours of roads in Singapore.

The proposed $G_{route}$ supports the following operators.

**Neighbor()** returns one-hop neighbors of a given vertex, where the users can specify to return incoming neighbors, outgoing neighbors, or both. This operator has $\mathcal{O}(1)$ com-


Figure 4.7: Visualization of the real route graph

plexity as we maintain adjacency lists for each vertex. Figure 4.6a shows both incoming and outgoing neighbors of vertex $v_2$.

**Insert()** inserts a student's ID into a specific vertex's "student_set" (duplicates can automatically be handled by set) and updates the inverted index accordingly. Insert() can be easily extended to process demands. Figure 4.6b shows the example of inserting S1's demands by sequentially calling Insert() on its potential pickup locations. In the meantime, a new record indexed by S1 is added to the inverted index.

**Remove()** incurs a reversed process and is typically used for ensuring one student can only be picked up at one vertex. Same as Insert(), Remove() works directly on vertices and thus is of $\mathcal{O}(1)$ complexity.

**Locate()** takes a student's ID as input and returns all the associated vertices by checking the inverted index. This operator is of $\mathcal{O}(1)$ complexity as we maintain an inverted index in the route graph.

**Estimate()** takes a shuttle route as input and estimates the ride time to school for each stops in the bus route. The ride time is estimated by a shortest path that connects all bus stops and thus is of $\mathcal{O}(V^2)$ complexity. This operator is efficient as the number of related vertices between two stops is small. Figure 4.6c shows an example output of Estimate().

**Allocate()** estimates the best vertex for each student to get on the bus given a certain

bus route. We make two reasonable assumptions, i.e., students will be willing to get on the shuttle as long as the total commute time of themselves can be reduced and they will always pick the bus stop that results in highest commute time reduction. After calling Allocate(), vertices in shuttle routes will sequentially check their "student_set". For each student in a "student_set", all traversed vertices will be retrieved by calling Locate(). The best vertex is the bus stop that results in the lowest commute time considering all the traversed vertices and the ride time. After that, other potential pickup locations of the student will be deleted from corresponding vertices by calling Remove(). This operator is of $\mathcal{O}(nm)$ complexity, where n is the number of bus stops and m is the size of "student_set". The operator is efficient as m and n are usually small compared with the number of vertices. Figure 4.6d gives an allocation on the route graph of Figure 4.5b: the student that origins from $v_1$ chooses to get on the bus at $v_2$; the student origins from $v_4$ will get on the bus at $v_4$; and the student that origins from $v_5$ sticks to his/her original path.

**Score()** calculates the beneficial score from an allocation. We define the beneficial score $\phi$ as a function that summarizes the gain and the cost of a route, where the gain is overall time saving for all students and the cost is total ride time for all buses.

$$\phi(routes) = time\_saving * \alpha^{ride\_time}, 0 < \alpha \leq 1 \qquad \text{(Eq. 4.1)}$$

where $\alpha$ is a user-defined tuning parameter to set the preference on the gain and cost for a certain application. A larger $\alpha$ indicates that more preferences are given to the reduction of students' commute time (e.g., $\alpha = 1$ means that we do not care about the system cost), while a smaller $\alpha$ puts emphasis on the system cost. Similar to many existing studies [31, 92] in route planning, we formulate the ride time into $\phi$ as it is highly related to the overall operating cost. Given an allocation, the overall time saving can be easily calculated via the number of students who remain in bus route vertices and the bus ride time can be estimated by Estimate(). Thus, this operator is of $\mathcal{O}(n)$ complexity, where n is the number of bus stops. For the allocation depicted in Figure 4.6d, Score() returns $\phi(\{v_4, v_2\}) = (1 * (20 - 6) + 1 * (10 - 3)) * \alpha^6$.

*A practical issue in graph annotation: the vertex ambiguity.* During the graph annotation, we aim to find a best matched vertex $v$ for each potential pickup location $p$ in trip profiles. The ambiguity stems from two facts: (1) pickup locations are far less in

granularity compared with vertices of the road graph. (2) pickup locations returned by Google Directions Service [43] are not always close to its corresponding vertices of the road graph generated by OpenStreetMap [86]. In our study, we observed that location proximity between $v$ and $p$ is not enough and sometimes misleading, especially when it comes to vertex-dense regions such as intersections, junctions, and overpasses. The key observation here is that Google Directions Service [43] usually reports turning points as they define the shapes of trajectories. Such a characteristic makes the graph distance of the correct vertex sequence the shortest. Thus, we propose finding the best match of each $p$ via anchor vertices that result the shortest path among all candidates. For each pickup location, we first extract vertices within a certain distance as its candidate set. Then the anchor vertices are selected by calculating the shortest path among the combination of candidates from different sets, where a Viterbi algorithm is applied to speed up the calculation. Finally, we complete the path between consecutive anchor vertices with corresponding shortest path and the best match of each $p$ is the nearest vertex along the completed path.

### 4.2.3 Graph-based Route Planning

With the proposed graph structure, we are now able to efficiently evaluate a specific bus route. However, given that enormous possible bus routes exist in city-wide route graph, brute force searching is still infeasible.

In this subsection, we first define the graph-based route planning problem and prove its NP-hardness. Then we propose a computationally feasible algorithm to ensure that we find a good route plan in limited time.

*Problem definition.* Given the route graph $G_{route} = (V, E, D)$ with $|V| = v$, expected number of routes $K$, a ride time distance $b$, we want to find a sequence of vertices $V' \subseteq V$, which maximizes the total beneficial score $\phi$ and fulfills two constraints: 1) $V'$ forms exact $K$ routes, 2) the ride distance of each route is no more than budget $b$. Mathematically, we formulate this problem as an integer programming problem, where we use binary variables $x_{ij}$ equal to 1 if and only if students in vertex $j$ is served by service route $i$.

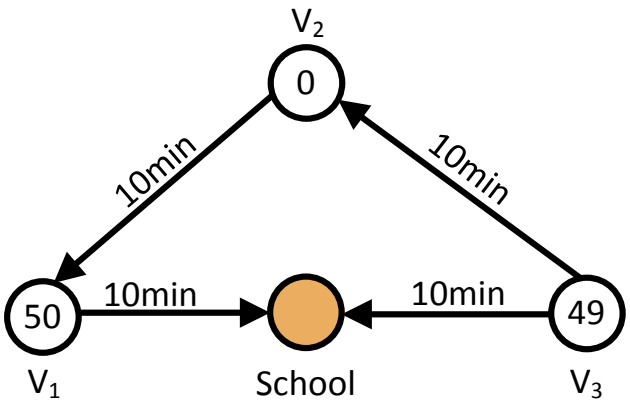



Figure 4.8: The shortsightedness of greedy expansion

$$
\begin{aligned}
\underset{N'}{\text{maximize}} \quad & \phi(N') = (\sum_{i=1}^{K}\sum_{j=1}^{V} g_{ij}x_{ij}) * \alpha^{C} \\
\text{subject to} \quad & \sum_{j=1}^{V} c_j x_{ij} \leq b \ (i = 1, ..., K) \\
& \sum_{i=1}^{K} x_{ij} \leq 1 \ (j = 1, ..., V) \\
& x_{ij} \in \{0, 1\} \ \ (i = 1, ..., K, \ j = 1, ..., V)
\end{aligned}
$$

where C stands for the total ride time of all routes, which equals $\sum_{i=1}^{K}(\sum_{j=1}^{V} c_j x_{ij})$, $g_{ij}$ and $c_{ij}$ denotes the saved time and the ride time respectively. The first constraint bounds the ride time of each route under budget $b$; the second constraint ensures that the students in one road segment are served by only one bus.

*NP-hardness*: Finding $K$ budget constrained last-mile bus routes with maximal beneficial score is NP-hard.

*Proof of the NP-hardness* We derive our problem by reducing a generalized assignment problem. We can view each road segment $v_j \in V$ being assigned to a route $r_i$ as a task being assigned to an agent. Each assigned task has a profit (i.e. students time saved) and a cost (i.e. the route time), while each agent has a budget (i.e. the route budget). The final decision set $V'$ is viewed as the task-agent assignment. In this way, for any instance of the decision version of the generalized assignment problem, we can find an instance of the decision version of the problem of finding K budget constrained last-mile

bus routes with maximum beneficial score by setting budget unbound, and their answers are the same. Thus, the generalized assignment problem is reducible to our problem, which completes the proof of NP-hardness.

*Greedy expansion algorithm.* We propose a greedy expansion algorithm to deal with the NP-hardness and plan the bus routes on the graph. It is based on an important mobility pattern of students, i.e., they eventually converge to the school from their homes scattered in the city. We initialize a solution at the school and progressively improve the solution by adding neighbor vertices as bus stops. In each iteration, the algorithm always chooses the neighbor vertex with the largest improvement on the beneficial score. If adding the neighbor vertices offers no improvement to current solution, the search terminates. This heuristic is computationally efficient, but it often results in a local optimum due to its greedy nature.

Figure 4.8 illustrates an example of this problem. For simplicity, we set the ride time of each edge equals 2 minutes, and tunning parameter $\alpha = 0.95$ in Equation Eq. 4.1. The search heuristic starts at the school and chooses $v_1$ as $\phi(\{v_1\}) = (50 * 8) * 0.95^2 = 361$ is larger than $\phi(\{v_3\}) = (49*8)*0.95^2 = 354$. When the greedy search continues to expand from $v_1$, it terminates because expanding to its only neighbor $v_2$ incurs a decrease in the beneficial score ($\phi(\{v_2, v_1\}) = (50 * 8) * 0.95^4 = 319$). Such shortsightedness stops the algorithm from searching further and finding the global optimal solution by including V3 ($\phi(\{v_3, v_2, v_1\}) = (50 * 8 + 49 * 4) * 0.95^6 = 438$)

*Tabu-based expansion algorithm.* We propose a tabu-based expansion algorithm that integrates the idea of tabu search [41] to explore the solution space beyond local optimality. Generally, tabu search starts with an initial route and searches for the best solution in a defined neighborhood of current solution. It then updates current solution by the solutions in the neighborhood and repeats the process until a certain termination condition is satisfied. It allows non-improving moves, but maintains a tabu list of forbidden moves to prevent cycling back to solutions that have already been visited. To leverage tabu search to perform route planning on our graph, we develop a tabu-based expansion algorithm that includes application-specific design, including the tabu list, neighbors and termination conditions. As depicted in Algorithm 1, our algorithm has the following stages:

**Initialization.** The algorithm initializes a subgraph, k initial routes and an empty

**Algorithm 1:** Tabu-based expansion

**Input:** $G_{route}$, $k$
**Output:** $route^*$

```
route ← Initialization(k)
while not Termination do
    neighbors ← ∅
    for r ∈ G_route.neighbor(route) do
        if r not in G_route.tabu_list then
            neighbors ← r
        end
    end
    candidates ← Evaluate(neighbors)
    if candidates.best.score > route.score then
        route ← candidates.best
    end
end
Return route
Function Evaluate(neighbors)
    candidates ← ∅
    for r ∈ neighbors do
        allocation ← G_route.allocate(r)
        c.score ← G_route.score(allocation)
        c.route ← r
        if new_score << old_score then
            G_route.tabu_list ← r
            continue
        end
        candidates ← c
    end
    return candidates
```

Tabu list. The subgraph is created with vertices that have a smaller "shortest_time" in respect to budget $b$. This reduces search space and ensures that all the planned routes within budget $b$. The initial routes start from the school.

**Tabu Iteration.** In each iteration, the algorithm first calculates the neighbor set of current solution. The neighbor set contains the routes that can be reached from the current route by adding an adjacent non-tabu vertex into current route or removing a non-tabu vertex from current route. If adding or removing a vertex results in a significant

drop in the beneficial score, we regard this vertex as a tabu vertex and append it to the "tabu_list" of the $G_{route}$. Tabu vertices are not allowed to be included in the neighbor set. The aforementioned shortsightedness is overcame by allowing non-improving searches until all the neighbors are tabu vertices. In each iteration, we update the route and score when the best route formed by neighbors has a larger score.

**Termination.** The algorithm terminates when either of the following two criteria is met, i.e., the maximum iteration number or no performance improvement in the last 5 iterations. The best solution identified in the whole search process is returned as the final solution.

## 4.3 Evaluation

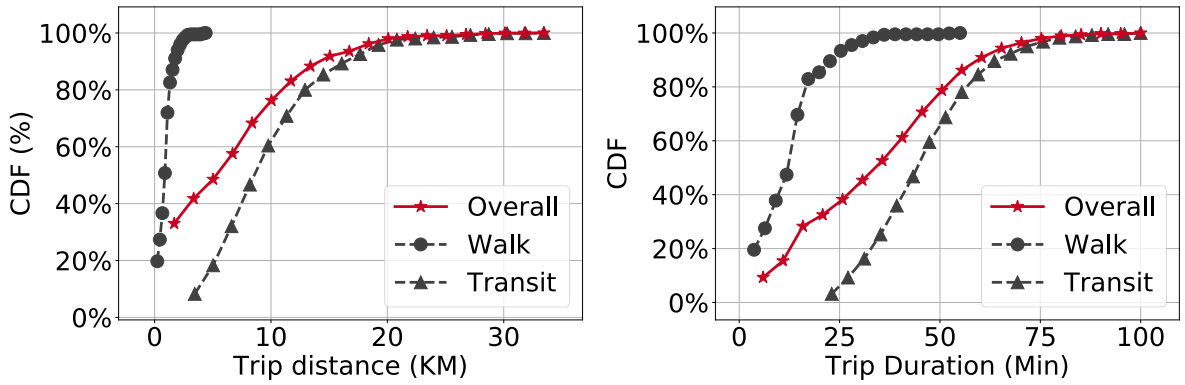
In this section, we conduct extensive experiments to evaluate our system using real-world data.

### 4.3.1 Experiment settings

We use the NSE data crowdsensed from 2809 students of 7 schools within a semester (i.e., from 11/04/2017 to 31/07/2017). During this four-month experiment period, one SENSg device is assigned to each student to continuously record the student's daily trajectories for one week. A total of 11236 trajectories that traversed $\sim$ 80k kilometers are extracted from the collected data. For each student, we use four-day data to perform our last-mile bus planning and use the remaining one-day data to evaluate the proposed system. We extract road networks of Singapore from OpenStreetMap [86], which contains 384653 road segments and 597446 edges.

We compare our results with the following methods.

- **Door-to-Door shuttles**. It offers minimum-distance routes by solving a capacitated VRP [61] formed by students' home locations.
- **Feeder shuttles** [132]. It first performs a KMeans clustering over students' home locations with the K having the largest Bayesian information criterion, and then offers minimum-distance routes by solving a capacitated VRP [61] formed by the centroids of obtained clusters.




(a) Trip distance distribution

(b) Trip duration distribution

Figure 4.9: NSE Mobility Statistics

- **Metro shuttles**. It offers routes that directly link the largest transportation hub and the school.

We set system parameters as follows: the parameter in objective function $\alpha$=0.995; the bus capacity is 100; the number of available buses is the minimum fleet size required in terms of the total number of students. As suggested by Figure4.9a, the overall trip distance of NSE students distribute almost evenly from a few kilometers to 20 kilometers. Since it will be hard and unfair to determine a service cut-off regarding trip distance, we adopt the original setting of serving all the students for Door-to-Door shuttles and Feeder shuttles..

All the algorithms are implemented and experimented with a HP Z440 workstation with 12 3.5GHz Intel Xeon CPU cores and 32GB memory.

### 4.3.2 Mobility Statistics of NSE Trajectories

In NSE data, after omitting students who traveled to schools by private cars, we have 37% of students who walked to school and 63% of students who commuted via public transits. We summarize their mobility statistics.

**Trip distance distribution.** Figure 4.9a summarizes cumulative distribution function of the trip distance. From the figure, it is clear that the majority of transit trips have a longer trip distance compared with walking trips, i.e., 90% of transit trips are shorter than 16.7 KM while 90% walking trips are under 1.8 KM. The overall average

trip length is 6.9 KM.

**Trip duration distribution.** Figure 4.9b illustrates cumulative distribution function of the trip duration, where 90% of transit trips take less than 63 minutes while 90% of the walk trips are within 23 minutes. This is because, upon a longer commute time, students are more prone to be driven to school by their parents. The overall average trip duration is 33 minutes.

**Trip general patterns.** We list three general patterns we found in NSE data: 1) Most of students (85%) arrive at school during 7:00 a.m. to 7:45 a.m. This proximity reveals the chance of finding last-mile bus routes that can benefit the majority of students and remain cost-efficient; 2) Most of the students' homes are near their schools, while a certain number of students live far from their schools. The proportion of students who live far away varies from school to school; 3) Students rarely change their choices of transportation in different days, which makes it plausible to use historical data for route planning.

### 4.3.3 System performance

Table 4.3 summarizes detailed information on the schools used in our evaluation. Due to the space limit, we list 4 schools as well as an overall summary of all schools. The other schools have similar statistics. For each school, we list information including the total number of students, the number of students who walk to school, the number of students who take public transits, the average distance and duration of all trips, and the average number of pickup locations. As shown in Table 4.3, the percentage of students who take public transits differs among schools due to the difference in public transits accessibility. Besides, some schools have students living relatively far away than other schools, resulting in longer trip distance and duration.

Table 4.4 summarizes the performance of four methods in terms of fours metrics: the average reduction on students' commute time (denoted as $\Delta t$), the total ride distance of bus services (denoted as d), the percentage of students that got served (denoted as %) and the beneficial score (denoted as $\phi$ and defined in 4.2.2). Figure 4.10 visualizes the shuttle routes planned for School A.

In Table 4.4 and Figure 4.10, we can see:

- Proposed system produce routes that outperform Feeder, Metro shuttles by 6.6×

Table 4.3: Summary of student commute patterns

| | School A | School B | School C | School D | Overall |
|---|---|---|---|---|---|
| # Students - Total | 448 | 418 | 472 | 292 | 2809 |
| # Students - Walk | 358 | 107 | 60 | 84 | 1052 |
| # Students - Transit | 90 | 311 | 412 | 289 | 1757 |
| Avg distance (KM) | 2.8 | 6.9 | 9.1 | 7.0 | 6.9 |
| Avg duration (Min) | 25 | 35 | 44 | 36 | 33 |
| **Avg # pickups** | **20** | **30** | **46** | **30** | **32** |

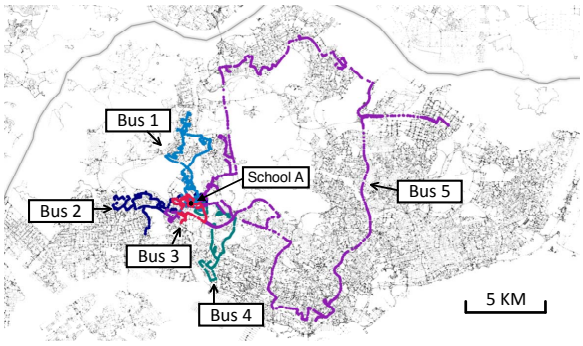
Table 4.4: Summary of overall performance

| | School A | | | | School B | | | | School C | | | | School D | | | | Overall | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\Delta$t | d | % | $\phi$ | $\Delta$t | d | % | $\phi$ | $\Delta$t | d | % | $\phi$ | $\Delta$t | d | % | $\phi$ | $\Delta$t | d | % | $\phi$ |
| Door-to-door | -7.7 | 278.7 | 100% | - | -15.9 | 529.8 | 100% | - | -16.3 | 727.4 | 100% | - | -35.5 | 431.0 | 100% | - | **-12.0** | **479.2** | 100% | - |
| Feeder | 2.7 | 157.0 | 100% | 1.2 | 1.2 | 375.7 | 100% | 0.2 | 3.7 | 198.2 | 100% | 1.4 | 3.6 | 187.7 | 100% | 1.4 | **2.9** | **209.5** | 100% | 1.0 |
| Metro | 1.3 | 0.8 | 21% | 1.3 | 1.5 | 0.4 | 42% | 1.5 | 2.6 | 1.5 | 17% | 2.5 | 2.2 | 1.0 | 26% | 2.2 | **2.4** | **1.2** | 24% | 2.3 |
| Our method | 5.1 | 4.5 | 96% | 5.0 | 4.0 | 5.0 | 96% | 3.9 | 9.4 | 7.7 | 73% | 9.0 | 7.1 | 3.5 | 85% | 7.0 | **6.8** | **6.1** | 87% | 6.6 |

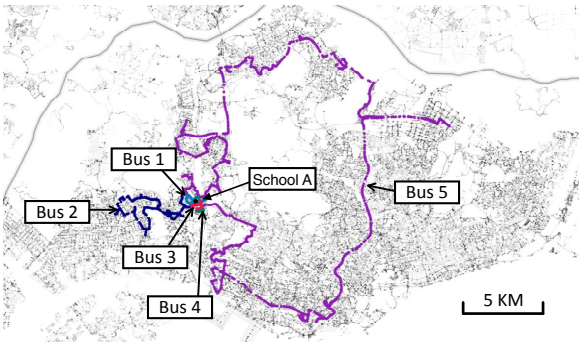
and 2.9× respectively in terms of overall beneficial score. This is due to its awareness of both students' real demands and existing public transits. By understanding how the students' transport demands are addressed by current public transits and locating the stops that can help them most in a global view, proposed system yields reasonable commute time reductions with relatively low ride distances.

- Door-to-door routes have the largest ride distances in all schools due to the total ignorance of existing public transits. The long ride incurs unnecessary long ride time for students who get on the bus on the early stage, which further lowers the reduction on all students' commute time. For Feeder, although the total ride distance is reduced by performing a clustering beforehand, the students, on average, need to travel ∼0.92 extra kilometers and 11.6 minutes to nearest service station due to its unawareness of road networks and existing transits. Consequently, the reduction to students' commute time is also limited.

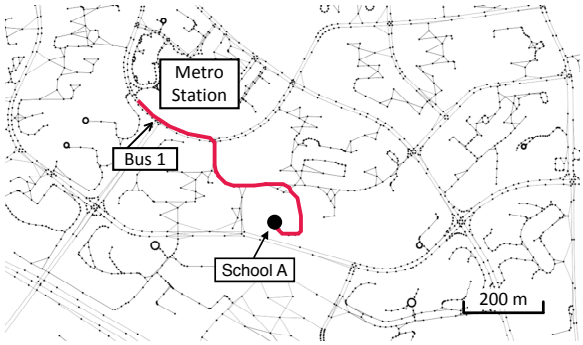
- Although fetching students from the largest transportation station nearby possesses a small ride distance, it can only serve a limited amount of students due to the fact that there are not much students' commute rely on one specific transportation station. The lack of understanding on students' demands limits the percentages of



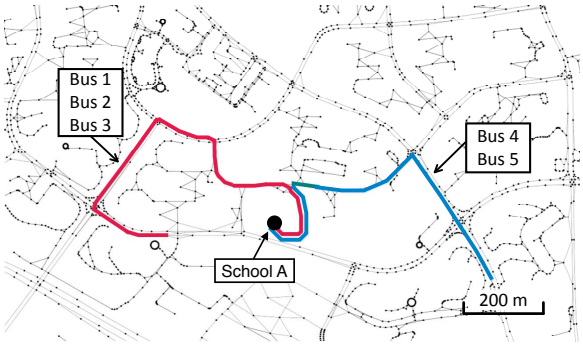

(a) Door-to-Door shuttle routes
map scale = 1:1000000
total bus routes = 278.7 km




(b) Feeder shuttle routes
map scale = 1:1000000
total bus routes = 157.0 km




(c) Metro shuttle routes
map scale = 1:25000000
total bus distances: 0.8 km



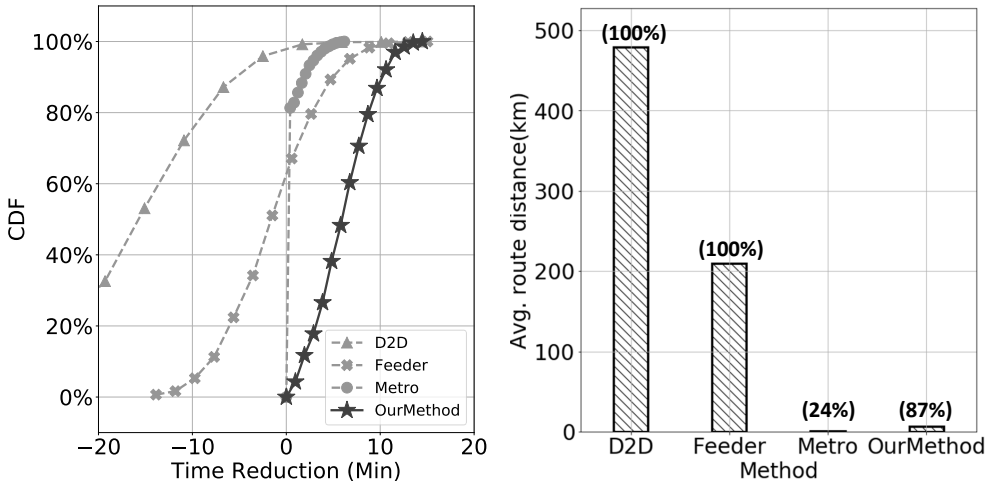

(d) Routes by our approach
map scale = 1:25000000
total bus distances: 4.5 km

Figure 4.10: Visualization of last-mile bus routes planned for School A

students who can benefit from metro shuttles (24% on average) and results in small reductions on students' commute time.

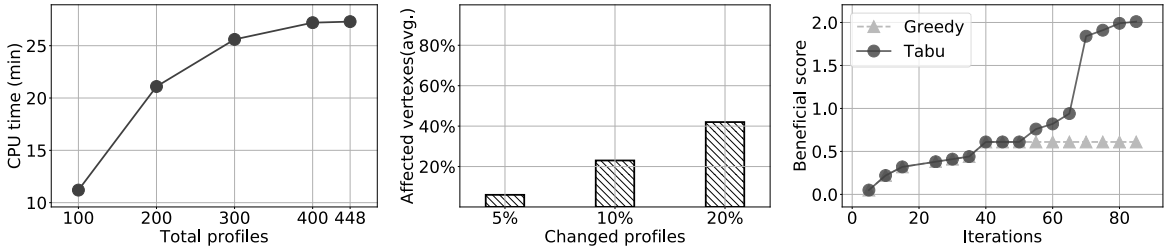
- Overall, the proposed system can benefit the majority of students (i.e., 87%) and offer a reasonable commute time reduction (i.e., 6.8 minutes).

Figure 4.11 shows the comparison on the overall gains and costs of routes suggested by different methods. Figure 4.11a illustrate the CDF of trip duration reduction of all students, where the right half of the figure summarizes the percentage of students who benefit from the system. As shown, most students receive no benefit from Door-to-Door (D2D) and Feeder shuttles due to the long ride time. Although the CDF of Metro shuttles locates at the right half of the figure, both the population of served students and the time reduction for them remain low. This is because Metro shuttles usually ride short




(a) Trip duration reduction (b) Average routes distance

Figure 4.11: Overall performance




(a) Computational efficiency (b) Scalability on profile changes (c) Effectiveness of Tabu expansion

Figure 4.12: Evaluation on the graph structure and Tabu-based expansion algorithm

distances. In contrast, routes suggested by our method offer a reasonable time reduction to most students. Figure 4.11b shows on average routes suggested by our method serve 87% of students with short route distances.

### 4.3.4 System Components

We further study the performance of the four key components in our proposed system separately.

**Trip identification**. It needs to identify homes and schools by classifying trip points and stay points. We manually labeled 77797 points from 20 randomly selected students, which form 68 valid trips with 1598 trip points. We compare proposed time-weighted

density clustering with both threshold and density based counterparts (introduced in Section 4.2.1) in terms of precision and recall. We tune parameters of above algorithms based on grid searches over sets of empirical values and set a fine distance threshold as 50 meters and duration threshold as 1 hour in threshold-based algorithm, the minimum distance as 200 meters and the minimum number of points as 10 in DBSCAN, and the minimum distance as 200 meters and the minimum duration as 1 hour in proposed method.

Table 4.5 shows that proposed method yields a significant improvement on precision for identifying trip points. It is because proposed method can 1) adapt to inconsistent localization accuracy by leveraging density-domain information and 2) tolerate sudden drifts by exploring time-domain information. For identifying home and school locations, our method outperforms DBSCAN on recall because the latter cannot identify home-/school clusters with insufficient sample points when a SENSg device enters its sleep mode.

Table 4.5: Comparison of trip identification algorithms

| | Trip points | | Home & School | |
|---|---|---|---|---|
| | Precision | Recall | Precision | Recall |
| *Threshold* | 4.8% | 92.4% | 19.3% | 100% |
| *DBSCAN* | 72.4% | 97.0% | 90.1% | 88.2% |
| *OurMethod* | 90.5% | 97.7% | 98.5% | 100% |

**Trip profiling**. We evaluate the proposed Google-based algorithm by the accuracy of sample-wise travel mode detection and the performance of pickup location extraction. We compare proposed method with the state-of-the-art algorithms leveraging decision tree and map matching (DT+MM), where mobility features including sample-wise distance, duration, speed and acceleration are calculated from consecutive samples. We manually labeled 11085 walking, 10075 transit and 7735 driving samples, where 70% of samples are randomly selected for training the DT classifier while the remaining samples are used for testing.

The results in Table 4.6 reveal that proposed method yields an overall 91% accuracy in detecting sample-wise travel modes, while decision tree classifier achieve only 44% accuracy for all travel modes. Since the decision tree classifier relies heavily on the

Table 4.6: The accuracy of travel mode detection

| | Walk | Transit | Drive | Overall |
|---|---|---|---|---|
| $DT + MM$ | 54% | 40% | 36% | 44% |
| $OurMethod$ | 91% | 93% | 89% | 91% |

Table 4.7: The performance of pickup location extraction

| | Precision | Recall | F1 score |
|---|---|---|---|
| $DT + MM$ | 41% | 63% | 65% |
| $OurMethod$ | 91% | 95% | 96% |

consistency of speed and acceleration, it suffers from large localization errors and sparsity of NSE data. As a consequence of inferior performance in travel mode detection, the DT+MM algorithm extract pickup locations with only 41% precision. Table 4.7 shows that the proposed method has a dominated performance in terms of the precision, recall and F1 score on extracting pickup locations.

**Benefits of the route graph.** We show the benefits of proposed graph data structure by comparing it with TSP-based route planning in terms of two aspects: computational efficiency and scalability on profile changes. We conduct a trace-driven simulation based on trips of 448 students from School A.

Figure 4.12a shows that graph-based route planning scales well in terms of CPU time as the number of profiles increases. We omit the CPU time of TSP-based route planning as it requires solving $20^{100}$ TSP instances for 100 profiles and this overhead increases exponentially as the number of profiles increases.

Figure 4.12b depicts that proposed graph data structure is highly scalable towards profile changes. The reason is that profile changes are reflected in the graph structure by modifying the attributes of related vertices, which is less cumbersome than updating distances with all the other points in TSP-based method.

**Tabu Expansion.** We demonstrate the effectiveness of proposed tabu expansion algorithm with a trace-driven simulation based on 448 students from School A. As shown in Figure 4.12c, the greedy algorithm stops searching too early due to its shortsightedness. Tabu-based expansion algorithm exploits a bigger search space by allowing non-improving moves and finds a route that has 3.3× higher beneficial score than the greedy route.

# Chapter 5

# Large-Scale Vehicle Trajectory Reconstruction with Camera Sensing Network

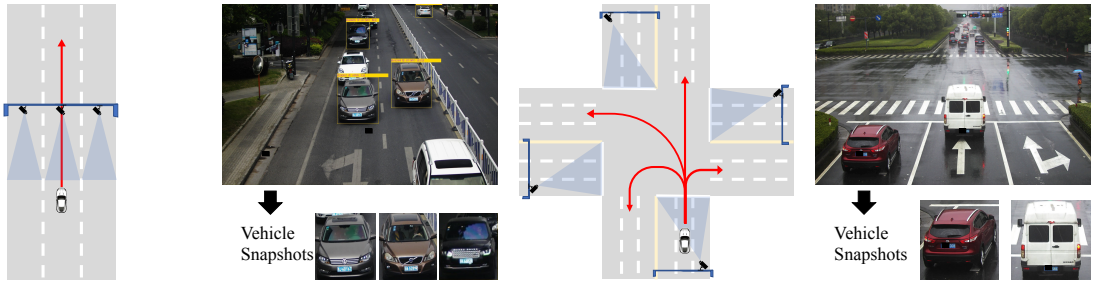
## 5.1 Sensing with Cameras

### 5.1.1 Why Vehicle Trajectories?

A vehicle trajectory is the path formed by the movement of the vehicle, usually represented as a sequence of chronologically ordered geolocations with timestamps. Vehicle trajectories offer fine-grained information for us to better understand mobility dynamics of individuals, communities, as well as the entire city, and thus foster a board range of applications. In the following we exemplify the potential use of vehicle trajectories with real world cases.

**Intelligent transportation systems.** Many intelligent transportation systems may be built with the knowledge of vehicle trajectories. For example, usage-based electronic toll collection (ETC) aims at precisely identifying the trajectory of each vehicle on the highway network and deducting fares based on the actual highway usage. Other applications include trajectory based road pricing for congestion management, traffic light control based on prediction of vehicle intention, and so on.

**Traffic analysis.** Traffic analysis provides key knowledge in understanding the mobility patterns of the city, and is essential in transportation research. Compared with sur-




(a) Front-viewed cameras (b) Front-viewed snapshots (c) Rear-viewed cameras (d) Rear-viewed snapshots

Figure 5.1: Deployment schemes and example frames of two common cameras in cities

veyed origin-destinations (ODs) that have been used in traditional transportation studies, vehicle trajectories provide a massive number of detailed vehicle mobility profiles. Recent studies have been using GPS tagged vehicle trajectories (e.g., those of taxi fleets) to conduct the analytical studies, e.g., OD analysis [136], travel time estimation [117], traffic modeling [77]), and with that to improve the transportation efficiency and resilience, e.g., traffic prediction [13], rider sharing [130], contingency planning [34, 58]. Obviously, a city wide complete vehicle trajectory dataset that contains general vehicle traffics can provide more complete traffic information and truthfully reflect the city mobility.

**Retrospective analysis for urban planning.** Vehicle movements also reflect the operations of the city. By analyzing the observed vehicle movements, researchers are able to discover functions of a city, e.g., functional regions [57, 129], value of real estates [35], facilitate the urban environment sensing, e.g., for air quality [139], for noises [140], trace the root of events, e.g., criminal footprints [114], congestions [59], improve municipal services and infrastructures, e.g., charging station/billboard placement [67, 135], bike reallocation [68], shuttle bus planning [14, 110], map completion [12, 66].

### 5.1.2 Camera Sensing Network

This work employs a camera sensor network for sensing vehicle mobility. A traffic camera is a video camera which monitors vehicles on a road. Traffic cameras are often mounted on overhead poles and aimed at one or more lanes within their field of view. Different from speed cameras which upon triggers take still pictures of much higher resolution, traffic cameras are purposed for observation and constantly take low-resolution videos. In urban

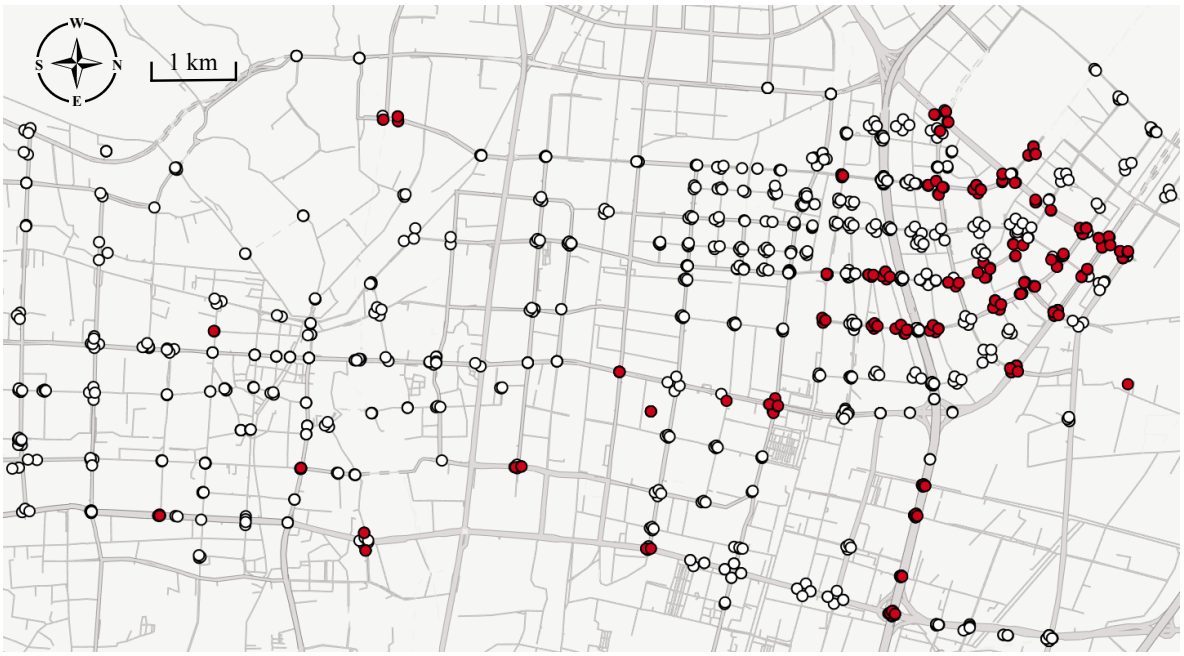


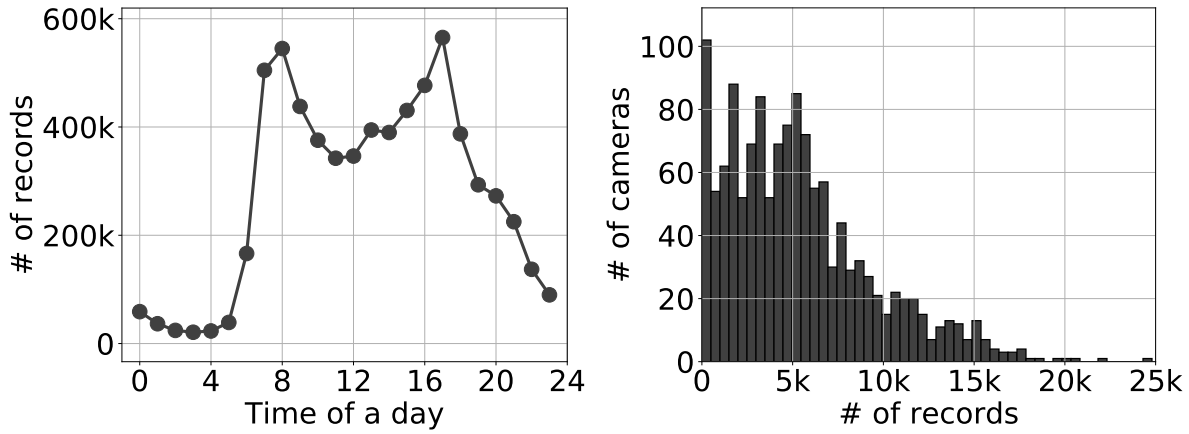

Figure 5.2: Camera distribution in urban area

areas, two types of traffic cameras are installed to capture the front face or the rear face of passing vehicles. Figure 5.1 depicts the two typical types of camera deployment, and shows example frames from the front-view and rear-view cameras. Front-view cameras are deployed for controlling access at important locations like entrances and exits of highways or key intersections of arterial roads. Rear-view cameras are used to monitor traffic violations (e.g., running a red light), and are deployed at intersections of surface roads, where both the vehicles and traffic light signals are recorded. The front-view cameras usually provide higher quality video frames than the rear-view cameras as a front-view camera is often focused on one lane. Beside that, we also observe differences of video quality across individual cameras.

**Infrastructure and data.** We investigate a central area in Hangzhou City, with observations from a network of 1342 traffic cameras that provide coverage of 66 $km^2$. Figure 5.2 visualizes the road network in the area and the locations of the traffic cameras. The front-view cameras are denoted in red and they are mainly distributed on the right of the map where locate highways and arterial roads that lead to the airport. The rear-view cameras are denoted in white and they scatter across road intersections over the entire area.

Table 5.1: A record of vehicle snapshot database

| Field | Value |
|---|---|
| Camera ID | 33010900001320000001 |
| Camera Location | The intersection of Shixin Road and Jinhui Road |
| Camera Direction | EAST |
| Camera Viewpoint | Rear |
| Entering Time | 2019-03-06 07:01:07 |
| Exit Time | 2019-03-06 07:01:10 |
| Snapshot | An Image of the detected vehicle |
| Lane directions | Right turn |




(a) Records over time

(b) Records over cameras

Figure 5.3: Statistic description

We perform a vision-based vehicle detection and tracking algorithm [72] with the video frames collected from the traffic cameras, and obtain snapshots each of which pictures an individual vehicle like those shown in Figure 5.1b and Figure 5.1d . The snapshot records are structured and each vehicle snapshot is associated with the time, location, lane, as well as other relevant information. Table 1 gives the information contained in an example snapshot record. Camera direction is the cardinal direction at which a camera is aimed. Entering and exit time are extracted by the vehicle tracking algorithm [72] referring the time when a vehicle enters and exits the camera's field of view. Lane directions refer to permissible driving directions of the lane where the vehicle is. Permissible driving directions include left turn, right turn, u turn, straight and their combinations.

On average, ∼7.2 million vehicle snapshots generated from the 1342 cameras every day. In this paper, we specifically report the results from the data collected on 2019-03-06, which contains 7206500 snapshot records in total. Figure 5.3 presents the data statistics in time and across cameras. Figure 5.3a plots the number of vehicle snapshots generated at different time of a day. It shows two peak hours (i.e., 8 am and 5 p.m.), which accords with the local traffic demands. Figure 5.3b plots the distribution of the number of snapshots generated by the 1342 cameras. 90% of the cameras generate fewer than 10k snapshots, and the majority generate 2k-6k snapshots a day.

**Data incompleteness.** The data as presented in Figure 5.3 suggests the fact that the snapshot data are spatial-temporally incomplete, and that is due to two main reasons. First, not all road intersections are covered by traffic cameras. The area depicted in Figure 5.2 contains 334 major intersections and only 190 of them (57%) are covered with traffic cameras. Second, not all traffic cameras function properly at all time. As Figure 5.3b shows, although the majority of cameras provide a proper recall of over 2k records per day, a significant portion of them (¿10%) generate fewer than 1k records, attributed to a few hardware issues (e.g., frame skipping, interrupted power, transmission loss) and software issues (e.g., failures in vehicle detection, corrupted images).

**Data inaccuracy**. The problem of data incompleteness is exacerbated by the inaccuracy introduced when extracting vehicle identities from the snapshots. The license plates are strong identifiers for vehicles, which is heavily relied on by state-of-the-art license plate recognition algorithms. The license plate contained within each captured snapshot however is often small and subject to conditions like blurred plates, dusty plates, obscured plates, incomplete plates, bad lighting, which introduces errors. Figure 5.4 (upper) gives examples with those imperfect conditions. Visual appearance is another useful clue used in identifying the vehicle, which however may also introduce errors due to the variance in image qualities and environment conditions, e.g., resolution, lighting, poses, and viewpoints. Figure 5.4 (lower) gives examples showing how a same vehicle may look different across cameras, time, and environment.

### 5.1.3 Uncertainties in Vehicle Identification

The incompleteness and inaccuracy of the camera sensing data lead to uncertainties in vehicle identification. We illustrate the cause of identify uncertainties, and quantify its

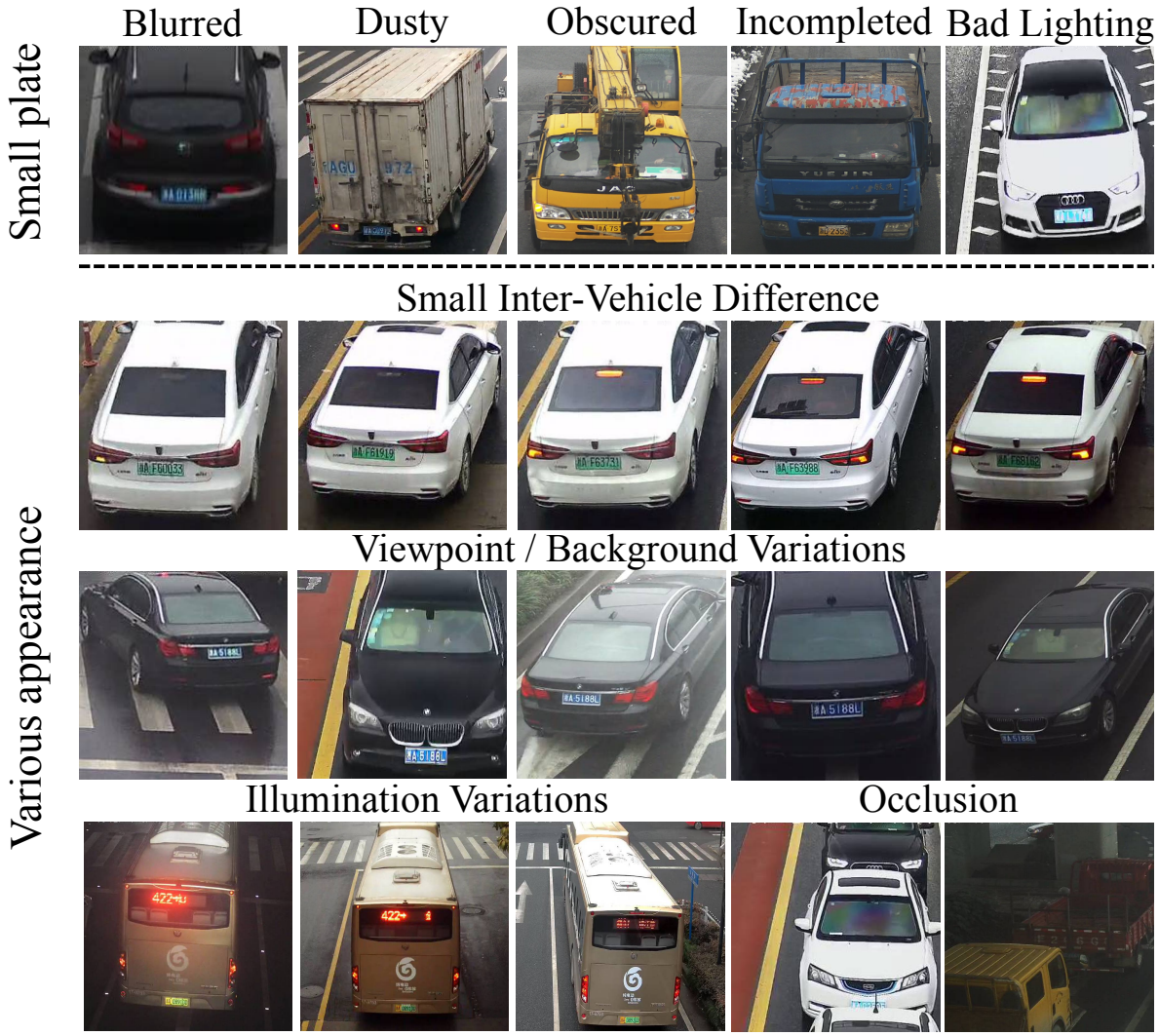
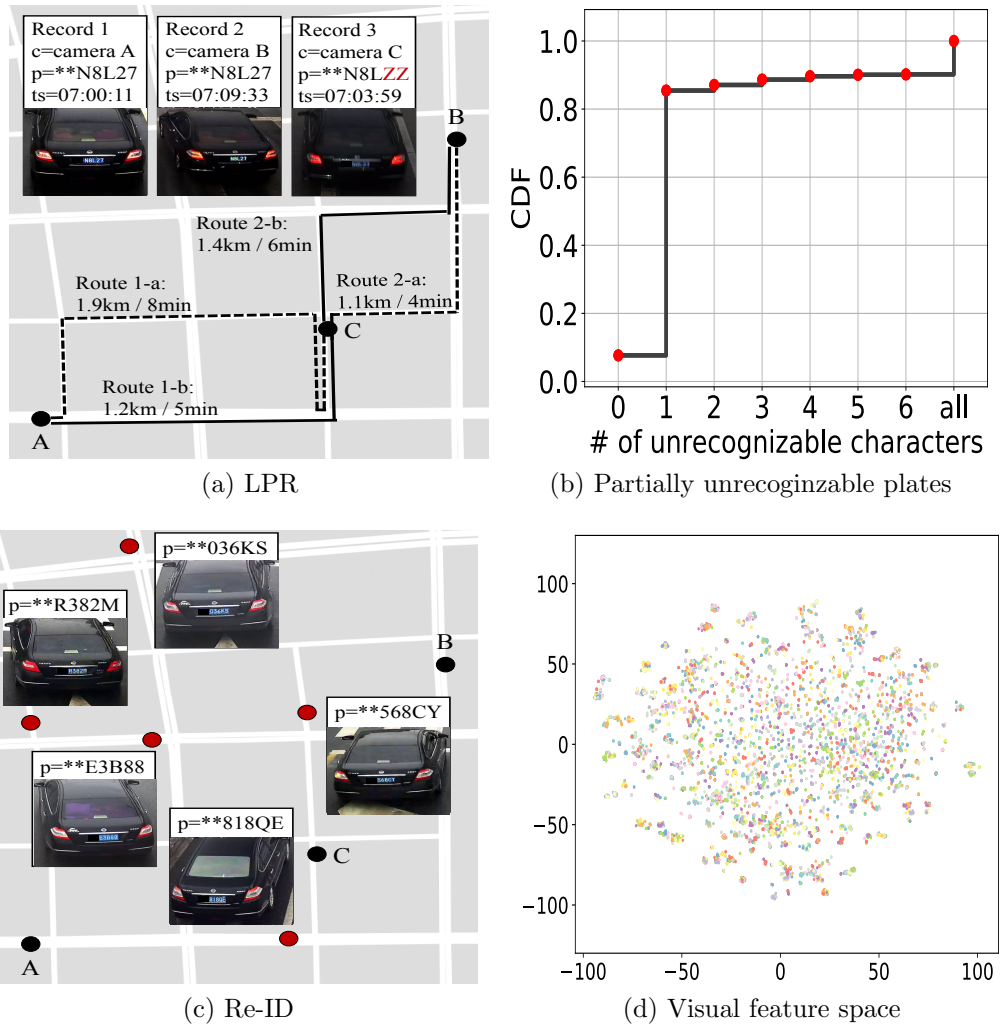



Figure 5.4: Identity uncertainties of vision-based solutions

impacts on state-of-the-art vision based solutions - LPR based on vehicle plates and Re-ID based on vehicle appearances.

**License Plate Recognition (LPR).** A LPR algorithm (usually a deep CNN-based model) can be employed to recognize plate characters for the vehicle in each snapshot. Snapshot records with the a same plate number are grouped as observations of the trajectory of a same vehicle. A most probable moving path that connects locations of consecutive observations with time and distance constraints can thus be inferred.

The accuracy of LPR based solution however is subject to accurate recognition of plate characters. Most existing LPR approaches assume high-quality input, normally




(a) LPR

(b) Partially unrecoginzable plates

(c) Re-ID

(d) Visual feature space

Figure 5.5: The Impact of identity uncertainties for LPR and Re-ID solutions

high-resolution images with decent lighting conditions, which is common in applications like automatic parking payment system where a front camera is dedicated to capturing car plates in short distance. Traffic camera snapshots are subject to a variety of common imperfections as Figure 5.4 (upper) shows, which make LPR error prone.

Inaccurate snapshot records of a same vehicle result in high uncertainty in inferring its trajectory. Figure 5.5a presents a typical example from what we experience. Due to LPR error, record 3 from camera C is wrongly recognized as a different plate number, and that leads to uncertain route candidates from camera A to camera B. Although the time and distance constraints may help remove such uncertainties, the efficacy has a limit, and

that is further impaired when the observations become sparse due to both LPR errors as well as insufficient camera deployment.

Figure 5.5b summarizes the LPR errors tested from a set of 2610 vehicle snapshots that we collect, and the result shows that less than 10% of the snapshots are correctly recognized. For those incorrectly recognized, over 75% contain only one unrecognized character, and 12% are completely unrecognizable.

**Vehicle Re-Identification (Re-ID).** Conventional vehicle Re-ID algorithms examine the vehicle snapshots in a dataset and rank them according to their similarity with a specific query snapshot. Usually, a feature extraction algorithm (e.g., a deep CNN-based model) is employed to extract a vector of global visual features from each snapshot. The vectors of different snapshots are compared to determine their similarity (e.g., cosine similarity). A predefined threshold is used to select those with sufficient similarity with the query snapshot. They are believed to belong to the same vehicle and used to reconstruct the vehicle trajectory.

As we illustrate in previous section, however, vehicle appearances are unreliable features in identification. The examples in Figure 5.4 (lower) suggest that 1) the same vehicle observed from different camera views may look different due to the variance in image qualities and environment conditions, and 2) different vehicles may appear similarly in visual features like colors and shapes. Such a problem is prevalent with large dataset of millions of snapshots. Figure 5.5c depicts that in the same example, if we apply Re-ID, five extra snapshots of other vehicles taken from different locations may appear similar, and the inclusion of them will severely mislead the trajectory reconstruction.

We apply a standard Re-ID algorithm [74] to examine the visual features of 8945 snapshots collected from the trajectories of 1746 vehicles. Figure 5.5d visualizes the feature vectors (each a sample point projected to a 2D space) of those snapshots, where the samples of the same vehicle are denoted in the same color. It is clearly shown that samples of different vehicles are tangled together and hard to distinguish solely from their visual features.

## 5.2 VeTrac Design

To tackle the identity uncertainties due to incomplete and inaccurate observation from the camera sensing network, we propose VeTrac, which exploits the mobility dependency and embeds that in a graph convolution process in order to reduce such uncertainties. VeTrac employs a multi-dimensional similarity (MDS) block to combine estimations from different aspects and gain extra confidence on vehicle identification (§5.2.1). A graph convolution network (GCN) is developed to capture complex correlations among vehicle snapshots and cluster the snapshots into different vehicle identities (§5.2.2) . The vehicle trajectories is produced based on snapshots belonging to different vehicle identities. VeTrac applies an iterative self-training process to further incorporate the mobility dependency into the GCN (§5.2.3) .

### 5.2.1 Multi-Dimensional Similarity (MDS)

VeTrac employs a multi-dimensional similarity (MDS) block to combine the similarity estimations for any two vehicle snapshots from three aspects, including LPR texts, vehicle appearance, and mobility causality. The following detail each of the three similarity estimators.

**LPR similarity.** Figure 5.5b suggests over 90% of the LPR texts contain errors with the traffic camera snapshots. The figure however also suggests that ∼80% of those only contain 1-2 unrecognized character, which means substantial portion of the LPR text can provide information on vehicle identities. In order to use that information, VeTrac employs a neural network model to quantify similarity between any pair of LPR texts.

Conventional LPR models assume fix-length inputs and adopt a CNN-based multi-headed network architecture, where a CNN-based backbone network (e.g., VGGNet) first extracts a global feature vector from the license plate image and uses a number of heads (i.e., fully-connected layers) to decode the feature vector into a fix-length sequence. Each of the head is trained to classify one character in the license plate image into one of many predefined characters. When a mismatch of plate length occurs (which is common with the traffic camera snapshots under the imperfect conditions like exemplified in Figure 5.4), the model wrongly segment the feature vector and thus introduces uncontrollable uncertainties to recognize the license plate.

Instead of the all-or-none strategy, VeTrac targets at recognizing only readable characters by convolutional recurrent neural network (CRNN) [103]. Originally introduced for recognizing text sequence in optical character recognition, CRNN supports variable-length recognition because it combines CNN with a bidirectional LSTM and CTC loss. CRNN consist of three types of layers: (i) Convolutional layers, which exact a feature sequence from the input license plate image. Different from CNN-based LPR models which aim at extracting a feature vector for each character, the convolution layers obtain a sequence of feature vectors, each of which corresponds to a region smaller than a character. (ii) Recurrent layers, which predict a label distribution for each feature vector. The recurrent layers consist of multiple bi-directional LSTM cells, each trained to predict a label distribution based on the information of its own element as well as information from adjacent cells. The unrecognizable characters on the plate can be skipped in the model. (iii) Transcription layers, which convert the prediction of each LSTM cell to a text of readable plate characters.

With the CRNN-based LPR model, VeTrac is able to handle license plates that contain unrecognizable characters or have varied lengths. The similarity between any two LPR texts can thus be modeled by the following Equation Eq. 5.1.

$$P_{plate}(A, B) = 1 - 0.08 * ed(A, B) * (ed(A, B) + 1) \qquad \text{(Eq. 5.1)}$$

where $ed(A, B)$ refers to the edit distance of plate text A and B.

**Appearance similarity** VeTrac adopts a CNN module for extracting high-level visual features containing information of the color, type and make from the vehicle snapshots. Different from previous works [73] that use sophisticated semantic attributes (e.g., the number of doors, the shape of lights, texture), we focus on those most intrinsic attributes as they are more commonly available and robust across snapshots of various resolutions, viewpoints and lighting conditions as offered by the traffic camera network.. The CNN module consists of a feature extraction backbone and three independent heads for multi-attribute classification. The backbone employs a ResNet-18 network [47] to extract a 512-dimension feature vector, which is then independently fed to three branches. Each head is a fully-connected layer and trained to classify one attribute of the following:

- The vehicle color head classifies each snapshot into one of 10 colors, i.e., white, black, grey, blue, red, yellow, green, brown, purple and pink.

- The vehicle type head classify each snapshot into 9 types, i.e., sedan, SUV, bus, minibus, taxi , van, MPV, truck, minitruck.
- The vehicle make head classifies each snapshot into one of 96 brands, e.g., for example, Audi, Honda, Nissan, Toyota, Volkswagen.

The cosine distance of above feature vectors is calculated to quantify the similarity between any pair of snapshots according to Equation Eq. 5.2.

$$P_{app}(A,B) = 1 - \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \qquad \text{(Eq. 5.2)}$$

where A and B are the corresponding appearance feature vectors of two vehicle snapshots.

**Mobility similarity.** VeTrac models the similarity between two snapshots taken at different time and locations by the probability that a vehicle can travel from one to the other within the time constraint $t = t_{snapshot1} - t_{snapshot2}$. The lane direction information is specially considered to build more accurate mobility transitions.

VeTrac first extracts direct neighbors of each camera by considering the road connectivity and camera lane information. With that, a camera transition graph is built where each vertex represents a camera and each edge denotes a direct link between neighboring cameras (that corresponds to the road segments connecting the two cameras). Two attributes are specified for each edge: (i) The transition time of each edge is estimated by the map API provided by a popular vehicle navigation service provider in the country. (ii) The lane direction of each edge denotes the driving action (i.e., right turn, left turn, straight, or u turn) it takes to transit from the camera to its neighbor on the graph.

With the camera transition graph, possible routes and their estimated travel time can be examined by depth search. In such a way, the mobility model between each pair of cameras can be learnt, and represented by an inverse Gaussian distribution $IG(\mu, \lambda)$. The mobility similarity between two snapshots can thus be calculated by the following Equation Eq. 5.3.

$$P_{mob}(t|\mu,\lambda) = \left[\frac{\lambda}{2\pi x^3}\right]^{1/2} \exp\left\{-\frac{\lambda(t-\mu)^2}{2\mu^2 x}\right\} \qquad \text{(Eq. 5.3)}$$

where t is the difference in timestamps of two consecutive snapshots, and $\mu$ and $\lambda$ are learned parameters of inverse Gaussian distribution $IG(\mu, \lambda)$, which models the travel

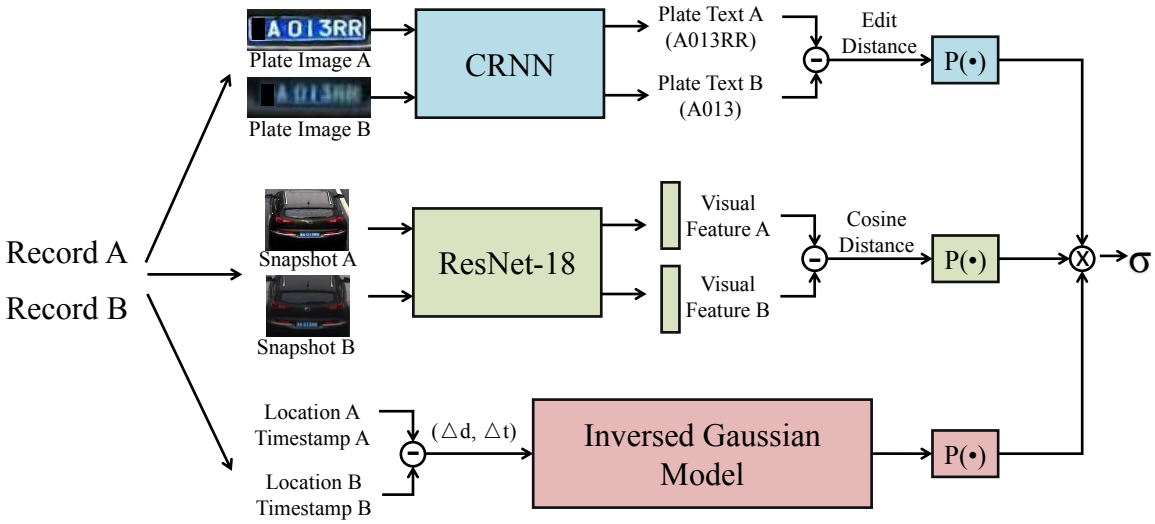



Figure 5.6: Multi-dimensional Similarity (MDS) block

time from the road segments of one camera to that of the other.

**The MDS score.** VeTrac combines the similarity estimation in a MDS block as depicted in Figure 5.6. The two snapshot records are examined in parallel by the three similarity estimators and their product is used to describe the probability of the two snapshots belonging to a same vehicle trajectory (Equation Eq. 5.4). As a result, MDS outputs correlation between any two snapshots by their similarities on all three aspects, and we call it MDS score.

$$\sigma_{A,B} = P_{plate}(A,B) * P_{app}(A,B) * P_{mob}(A,B) \qquad \text{(Eq. 5.4)}$$

### 5.2.2 Learning Vehicle Identities with Graph

MDS outputs a similarity score between any pair of vehicle snapshots. To maintain consistent identities across all the 7.2 million snapshots, VeTrac needs form clusters based on the pairwise MDS scores.

Query-based algorithms have been conventionally applied to retrieve similar snapshots for a given query snapshot. A key to its accuracy is to determine a proper threshold for similarity, which however is difficult in practice. A low threshold may lead to low precision while a high threshold may lead to low recall, due to the inherent variance in traffic camera observations. There exist non-negligible amount of hard cases with

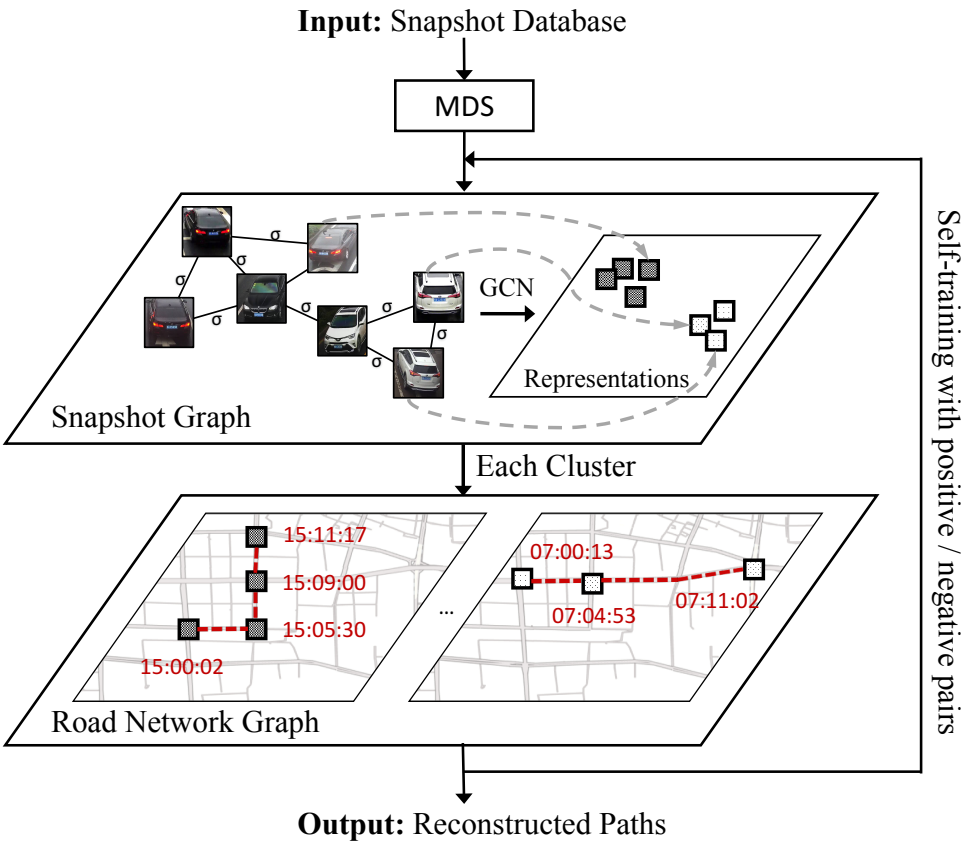



Figure 5.7: VeTrac: System overview

high intra-trajectory diversity or inter-trajectory similarity which fail the query-based algorithms.

VeTrac addresses such a problem by employing a graph structure, and modeling complex correlations among many snapshots rather than the pairwise similarities. By considering system-wise similarities, the identities of snapshots that fail in pairwise similarity assessment may be inferred from correlation of their similarity assessments with other commonly contacted peers. Errors introduced in MDS estimation may also be mitigated through a graph convolution process.

Figure 5.7 gives an overview of VeTrac workflow. The MDS scores are taken as input to a graph convolution network (GCN) containing all vehicle snapshots to examine. After a graph convolution process, a new identity representation for each snapshot is learnt and snapshots can then be clustered into individual vehicle trajectories based on their representations. In the following, we detail the GCN-based learning process in three

steps.

We detail our novel 3-step GCN-based learning pipeline, which includes: (i) constructing the snapshot graph with road connectivity and MDS block (ii) learning new identity representations with GCN (iii) generating vehicle trajectories based on learned representations

**Snapshot graph construction.** With the full dataset of vehicle snapshots, VeTrac builds a undirected graph $G_{snapshot} = (V, E)$, where a vertex set $V$ corresponds to all snapshots and an edge set $E$ denotes the MDS score between two snapshots belonging a same trajectory. With all estimated MDS scores between arbitrary pairs of snapshots, theoretically the graph $G_{snapshot}$ has an edge between any pair of vertices. Such a graph however will introduce quadratic complexity, i.e., $\mathcal{O}(n^2)$ where $n$ is the number of vertices, to convolution process, and thus not scale to our graph containing millions of vertices. In practice, VeTrac reduces the number of edges by removing those of large spatio-temporal spans. We define two vertices are k-hop connected if the two snapshots are taken from cameras within k-hop distance in the road network. Similarly we define two vertices are T-minute valid if the time difference between their timestamps is below T minutes. An edge is generated only for those k-hop connected and T-minute valid vertices, so the number of edges can be significantly reduced. In VeTrac system implementation, k and T are empirically set to 3 and 10. We assign the MDS score as the weight for each edge, which indicates the similarity between the two snapshot vertices that the edge connects.

**Learning identity representations.**. Graph convolution is performed on the snapshot graph so each vertex keeps updating itself to a suitable representation based on the knowledge it gains from its neighboring vertices.

Figure 9 depicts the developed GCN structure and the learning process.

In the input layer, for each vertex $v_i \in V$, a representation vector $x_i$ is initialized to represent its identity. To start with, the one-hot encoding technique can be used to assign each vertex a different random representation vector, e.g., a 1-by-n vector with its i-th element to be 1 and the rest to be 0 for $x_i$, where n is the number of vertices in $V$. In practice, we leverage the LPR results to initialize representations and that speeds up the convolution process. The rationale is that, the more a certain LPR text is observed across different cameras, the more likely that the LPR text is a correct one, so for a snapshot with a LPR text that appears more than three times in the entire dataset we consider

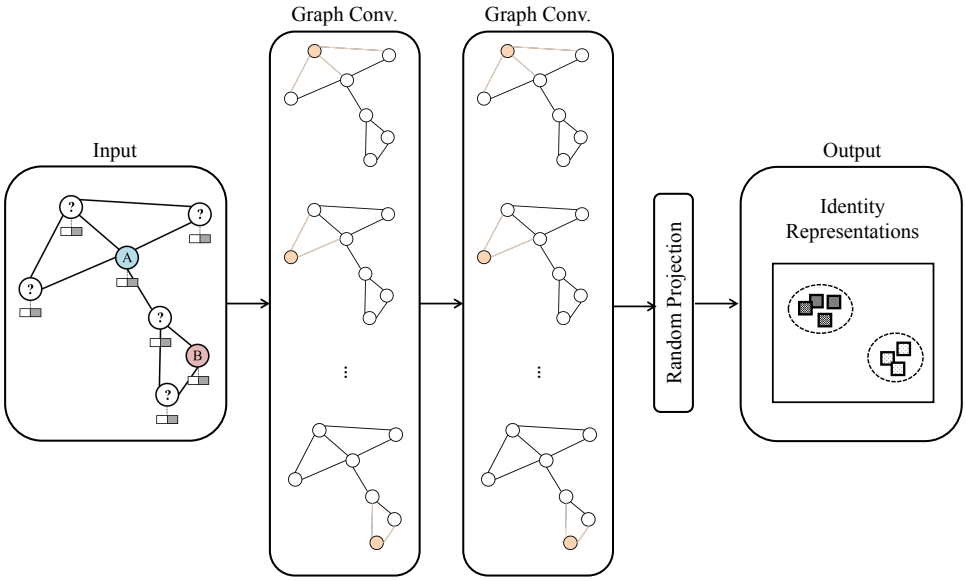



Figure 5.8: The proposed two-layer GCN units

the LPR text credible and thus label all involved snapshot vertices a same representation vector. The initial representation matrix $X$ of the graph can thus be constructed by combining representation vectors of all vertices, as indicated by Equation Eq. 5.5.

$$X = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \qquad \text{(Eq. 5.5)}$$

An adjacency matrix $A$ is also constructed with MDS scores as edge weights, as indicated by Equation Eq. 5.6. Each element in $A$ represents the probability of corresponding two records belongs to a same vehicle. We add self-loops to the snapshot graph by setting the diagonal of the adjacency matrix $A$ as 1. Since the graph only contains a significantly reduced edge set, the adjacency matrix $A$ is a sparse matrix with most elements set to 0. The sparsity of $A$ ensures the computational efficiency when performing graph convolution.

$$A = \begin{bmatrix} 1 & \sigma_{1,2} & \cdots & \sigma_{1,n} \\ \sigma_{2,1} & 1 & \cdots & \sigma_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n,1} & \sigma_{n,2} & \cdots & 1 \end{bmatrix} \quad \text{(Eq. 5.6)}$$

In graph convolution layers, the representation of each vertex is iteratively updated by aggregating its own representation and the representations of its neighbors. The aggregation is a normalized averaging with regard to edge weights. The propagation rule of graph convolution at each iteration can be expressed in matrix multiplication as indicated by Equation Eq. 5.7.

$$X_{i+1} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X_i \quad \text{(Eq. 5.7)}$$

where $D = diag(d_1, d_2, ..., d_n)$ is the degree matrix of $A$, $d_i = \sum_j a_{ij}$ is the degree of vertex $v_i$, and $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ performs symmetric normalization on the adjacency matrix $A$.

**From representations to identities.** A two-layer GCN unit (as indicated in Figure 5.8) is used to learn new identity representations. The two graph convolution layers complete the aggregation of updated identity information - in the first layer each vertex updates its representation with knowledge from its neighbors and in the second layer the updated representations are aggregated across the snapshot graph. In such a way, the vertices converge to proper representations and the complex correlations in the graph are encoded into the those representations. The snapshots taken from the same vehicle trajectories tend to agree while those of different tend to differ. Therefore, we can cluster the snapshots into individual trajectories by identifying snapshots of similar representations. Since the learned representation vectors are sparse, VeTrac first employs the random projection algorithm [8] to reduce the dimensionality to 512 and then performs HDBSCAN [11] to cluster vertices based on the cosine distance. The random projection algorithm and HDBSCAN are applied for their accuracy and efficiency in processing sparse vectors.

### 5.2.3 Self-Training with Road Network Graph

The GCN learns graph wide correlations and mitigates identity uncertainties from inaccurate pairwise similarity estimations. As a result, clusters of same vehicle identities are identified. For each cluster of snapshots, VeTrac is able to generate a maximum likelihood trajectory that connects them based on their timestamps.

The reconstructed trajectories, though greatly improved from those query based similarity matches, may contain errors in its precision (different vehicle snapshots contained in the trajectory) and recall (not all snapshots in the trajectory recalled). We investigate the performance of the GCN and do an error analysis with a small scale test with 8945 snapshots from 1746 vehicles. Over 70% of the generated clusters still contain errors mainly due to two problems: (i) 14% of them are incomplete clusters which contain only snapshots from the same vehicles but do not have full recall, and (ii) 85% of them are imprecise clusters which contain snapshots from more than one vehicles. As we introduce more graph convolution units in the GCN for thorough identity propagations, the former source of errors may further go down while the latter source of errors persist. That means the inaccuracy in identity clustering stems from the difficulty in separating negative pairs of snapshots (negative samples) compared with aggregating positive pairs of snapshots (positive samples) in the clusters.

VeTrac employs the knowledge of mobility dependency within the road network to address the issue. Specifically, as depicted in Figure 4.3 an iterative self-training process is introduced, which detects positive and negative samples from the reconstructed trajectories and feeds the information back to the GCN to minimize identity propagation from those negative samples. The self-training process enables the GCN in instilling trajectory-level mobility into the representation learning process. Three steps are involved in building the self-training loop.

**Build the road network graph.** VeTrac builds a road graph $G_{road} = (V, E)$ from the road networks. A road graph $G_{road}$ is a directed graph, where a vertex set $V$ includes all road segments each represented as a vertex, and an edge set $E$ denotes the linkage and physical directions between adjacent road segments. The information about each road segment (i.e., coordinates of its two ends, name, length, category, accessibility for vehicles) is extracted from OpenStreetMap, which is then used to derive the linkage between adjacent road segments. The vehicle transition time on each edge is obtained

obtained from the third party vehicle navigation service [38]. VeTrac maps the snapshot graph $G_{snapshot}$ to the road network graph $G_{road}$, and each snapshot is mapped to its corresponding vertex in $G_{road}$. The transition time from one snapshot to another can thus be estimated via shortest path search on $G_{road}$.

**Detect positive and negative samples.** For a specific snapshot, its similar snapshots that has a small cosine distance can be retrieved. By modeling trajectory-level mobility, VeTrac examines whether those snapshots constitute one or more feasible trajectories. Specifically, the snapshots are first mapped into the $G_{road}$. With their timestamps and lane directions (see 5.1), possible trajectory candidates are enumerated on $G_{road}$. The feasibility of a trajectory is then calculated based on transition probabilities regrading time and distance between every two consecutive snapshots. The transition probability of two snapshots is modeled by an inverse Gaussian, as indicated by Equation Eq. 5.3. The overall feasibility of a trajectory is derived as the product of all consecutive transition probabilities and regularized by the total number of concerned snapshots. An empirical threshold in the range of 0.65-0.85 can be set to judge the feasibility (0.8 is set in VeTrac system implementation, though the performance in insensitive to the exact value). Based on that, VeTrac labels all pairs of snapshots on $G_{snapshot}$. Each pair of snapshots from a feasible trajectory is considered a positive sample, and that from infeasible trajectory is considered a negative sample.

**Iterative graph convolution with positive and negative samples.** With labeled positive and negative samples, VeTrac adjusts the adjacency matrix $A$ in GCN. For each positive pairs of snapshots, an edge is added if there exists none in the original $G_{snapshot}$, and the MDS block is invoked to derive its weight (i.e., MDS score). For each negative pair of snapshots, the edge if exists in $G_{snapshot}$ is removed. After the adjustment, VeTrac reruns the graph convolution process in its GCN and updates the identity representations. The self-training process is iterative and terminates when all newly obtained samples become positive.

After the iterative graph convolution and self-training process, VeTrac gains clean and confident identity representations, and with that reconstructs feasible vehicle trajectories.

## 5.3 Evaluation

In this section, we present large-scale trajectory reconstruction results on a real-world datasets (§5.3.1), compare VeTrac's performance with state-of-the-art methods under various scenarios (§5.3.2), and analyze VeTrac's components and efficiency (§5.3.3).

### 5.3.1 Experimental Setups

**The dataset.** We investigate the central area of Hangzhou Xiaoshan district, which has 1342 functioning cameras covering an area of 66 $km^2$. On average, there are about 7.2 million vehicle snapshots generated every day from 1342 cameras. We collect one-day video. After processing the video (see §5.1.2), we obtain 7206500 snapshots, which makes up approximately 150 GB.

**Experimental setups.** We implement VeTrac on a server equipped with Intel Xeon E5-2682 2.50GHz CPU, an NVIDIA RTX 2080Ti GPU (32 GB RAM). After loading all the data, the offline process takes up approximately 3 days. VeTrac takes approximately 16 hours to reconstruct all vehicle trajectories.

**Comparison schemes.** To the best of our knowledge, none of existing works directly reconstruct trajectories of general vehicles in a city. We thus borrow key methods of three different principles and adopt them to reconstruct vehicle trajectories from snapshots:

- **License plate recognition (LPR) [105].** This scheme represents a common practice to obtain vehicle trajectories from camera snapshots. A DNN-based license plate recognition model (described in §5.1.3) is first used to extract a license plate from each snapshot, based on which snapshots are then grouped to assemble trajectories. We implement this scheme by employing VGG-16 [105] to recognize license plates, ordering snapshots of a same license plate according to their timestamps and forming vehicle trajectories by linking consecutive snapshots with the shortest path on the road networks.

- **Vehicle re-identification (Re-ID) [74].** This scheme represents a wide range of image-based re-identification methods (described in §5.1.3), which rank snapshots in the database according to their visual similarity towards a query snapshot. Considering high false-positive rate of only using visual features, we adopt a progressive vehicle Re-ID pipeline as suggested by [74]. We first employ an visual-based coarse

filtering to select candidate records, where the similarity cutoff is set as 0.8 of the highest similarity score. Then we use plate and spatiotemporal information to filter out false-positives in candidate records and form vehicle paths by linking consecutive candidate records with the shortest path on the road networks.

- **History based route inference system(HRIS) [138].** This scheme represents methods that aim to reduce uncertainties of low-sampling-rate trajectories. Instead of looking into camera data, this scheme assumes that trajectories with a same origin and destination (OD) usually have similar route choices and thus rebuild uncertain pieces of a trajectory by referring other vehicles with the same OD. We implement this scheme by taking the results of LPR as input and assign the most popular routes sharing same ODs as final trajectories.

**Evaluation metrics.** We define three metrics as follows.

- *acc* to measure accuracy of reconstructed trajectories. We regard a reconstructed trajectory as a correctly reconstructed trajectory if it is identical to its ground truth. *acc* is the ratio of correctly reconstructed trajectories among all the trajectories.

- $p$ to measure precision of a cluster. A large $p$ indicates a large percentage of records in a cluster belong to the correct vehicle. When a cluster only have records from the correct vehicle, $p = 1$.

$$p = \frac{\#cluster_{correct}}{\#cluster} * 100\% \qquad \text{(Eq. 5.8)}$$

  where $\#cluster_{correct}$ is the number of snapshots that belongs to the correct group and $\#cluster$ is the total number of snapshots in the cluster.

- $r$ to measure recall of a cluster. A large $r$ indicates a large percentage of records belonging to the correct vehicle are included in a cluster. When all the records from the correct vehicle are contained in a cluster, $r1$

$$r = \frac{\#cluster_{correct}}{\#vehicle} * 100\% \qquad \text{(Eq. 5.9)}$$

  where $\#cluster_{correct}$ is the number of snapshots that belongs to the correct group and $\#group$ is the total number the group have.

### 5.3.2 Experimental Results

We access the quality of reconstructed paths by emulating two common use cases of VeTrac:

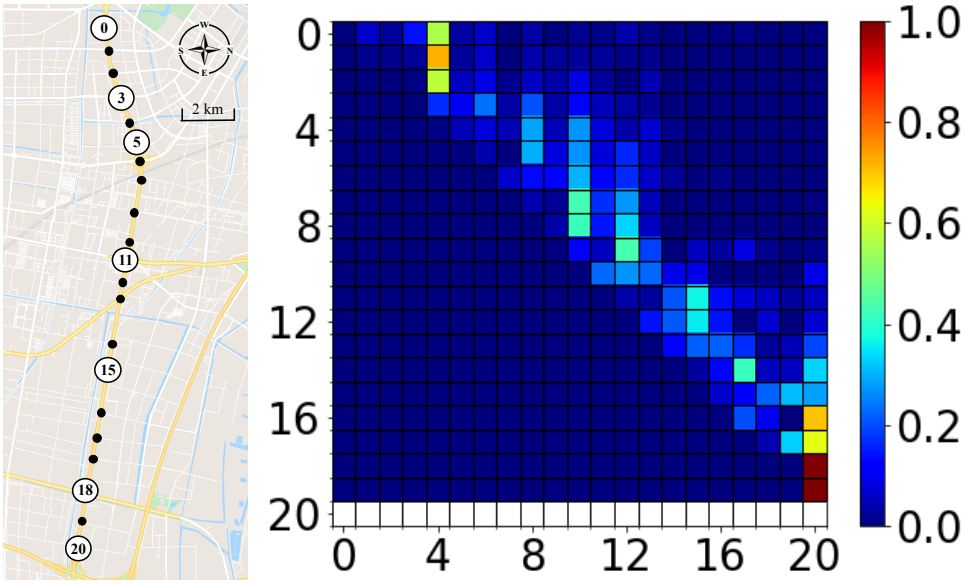
- **Highway: completed ground truth** We select a highway dataset from the data of a local airport expressway. As visualized 5.9a, the highway consists of 21 interchanges from west to east, where vehicles can either enter or exit the expressway via ramps in each interchange. A total of 54 cameras are deployed to monitor traffic in those interchanges. We collect morning-rush-hour data (i.e. from 7 a.m. to 9 a.m.) and get 8945 snapshots. Since the expressway has simple road networks and relatively isolated traffic, we are able to manually label the true identity of all obtained snapshots, which contains 1746 vehicles. With the completed ground truth, we evaluate the system performance under different situations by manipulating the original data.

- **Urban: large-scale datasets and complex road networks.** We build the urban dataset by selecting morning-rush-hour data (i.e., from 7 a.m. to 9 a.m.) from the original dataset, which consist of 1098467 snapshots from 1342 cameras. Due to the difficulty in labeling a dataset of such a scale, we carefully label the ground truth of two representative subsets: *public transits* and *private vehicles*. For *public transits*, as shown in Figure 5.13a, we collect 125 trajectories of 3 bus lines by linking the routes and schedules released by bus company and snapshots captured by the camera network. For *private vehicles*, we randomly select 300 snapshots, manually labeled their identities by referring related snapshots and then identified a most likely trajectories for each snapshot with regard to road networks.

#### Highway

Figure 5.9b visualizes the true origins and destinations of the trajectories of 1746 vehicles. As shown, the vehicles' ODs distribute evenly across 21 interchanges. Most of the vehicles (more than 80%) passed 4 or more interchanges before they exited the highway.

We first show a representative example of reconstructed trajectories by VeTrac, LPR, Re-ID and HRIS, respectively. Figure 5.10a shows the proposed system VeTrac can still yield the correct trajectory, even though some license plate are wrongly recognized (i.e.,




(a) Highway (b) Origins and destinations

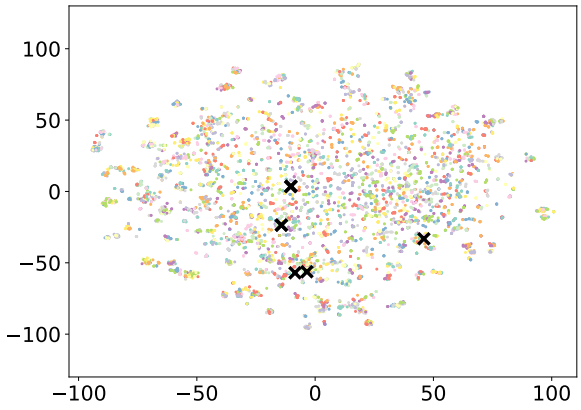
Figure 5.9: Highway: visualization of origins and destinations

snapshots captured by camera ⑦, ⑨ and ⑪) and some snapshots are visually differs with the rest (i.e., snapshots captured by camera ⑥, ⑩ and ⑪). In contrast, LPR are vulnerable to plate errors even if the error is small, while Re-ID fails to recall snapshots with a different viewpoint due to large difference in visual features. As for HRIS, since the majority of trajectories start from camera ⑤ end at camera ⑧, HRIS also cannot rebuild the correct trajectory even in a simple road networks.

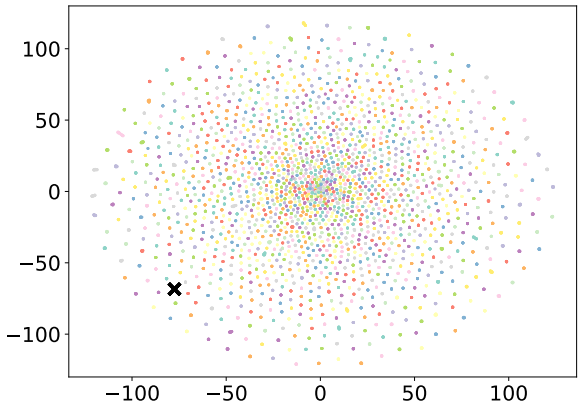
We further present the statistical results of reconstructed trajectories. We visualize the identity representation space of VeTrac and visual features in Figure 5.10c and Figure 5.10b by TSNE algorithm [79], where snapshots of same identities are marked as one same color. As shown, the seven snapshots are clustered together in the representation space of VeTrac, but they separate far away in the representation space of visual features. Comparing with the visual feature space, in the representation learned by VeTrac, snapshots with different appearances but a same identity are located more closely with each other (average intra-class distance is 1% of that of visual features), while snapshots with similar appearances but different identity are separated more widely (on average, VeTrac introduce 0.3 false-positives to recall all related snapshots while visual features introduce 387.5 false-positives). With the high-quality representations, VeTrac can handle hard cases in later clustering.

| Camera | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|
| Ground Truth |  | | | | | | |
| VeTrac | √ | √ | √ | √ | √ | √ | √ |
| LPR | √ | √ | | √ | | √ | |
| Re-ID | √ | | √ | √ | √ | | |
| HRIS | √ | | | √ | | | |

(a) Comparison of reconstructed trajectories




(b) Visual representation




(c) VeTrac representation

Figure 5.10: Highway: An representative example. The black crosses in (b) and (c) denoted the locations of the above seven snapshots in the representation spaces.

Figure 5.11a shows the average accuracy, precision and recall of VeTrac compared to the baselines. Overall, VeTrac yields 98% accuracy, which outperforms all the baselines by at least 32%. For precision and recall, VeTrac achieve 99% recall at the cost of merely 1% decrease in precision. For LPR, although all reconstructed points are correct, it can not handle snapshots with wrongly recognized plates, which results in only 74.3% of the correct records are recalled. For Re-ID, although achieving a slight improving on the recall (86.2%), its precision also drops (74.3%) compared with LPR. HRIS also have relatively low performance (41.3% accuracy). This is because the route choice between similar ODs are not always the same, which contradict to the assumption of HRIS and limits the power of referring other trajectories.

Finally, to evaluate the robustness of VeTrac, we perform controlled experiments by

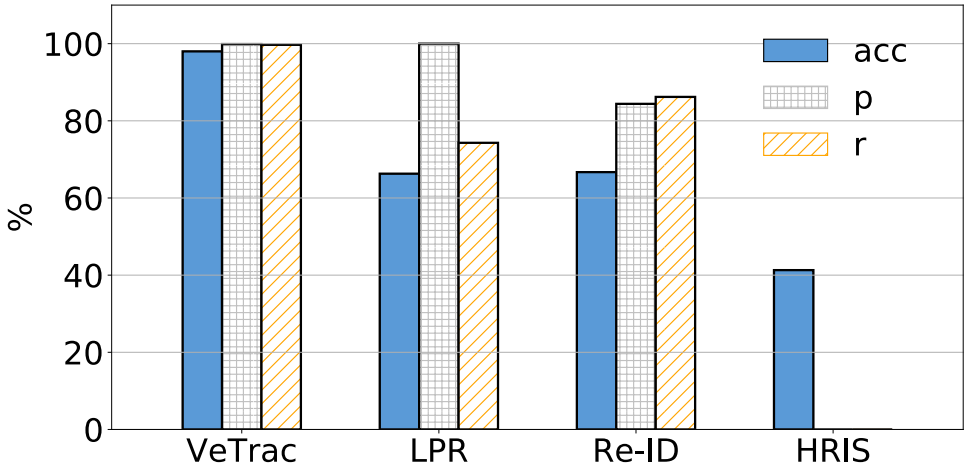
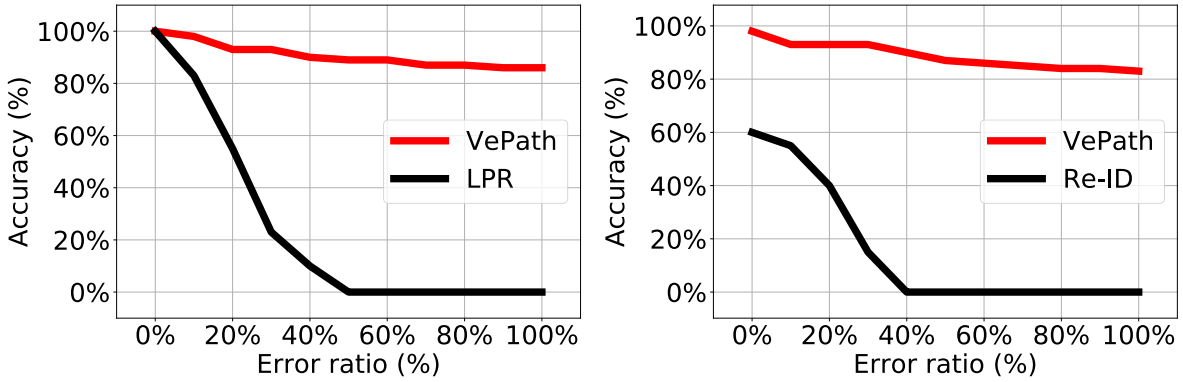



Figure 5.11: Highway: statistical evaluation

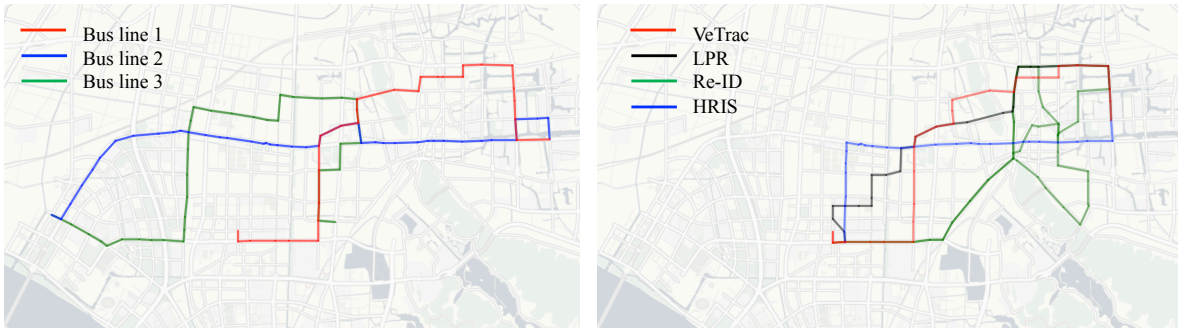



(a) Wrong LPR texts

(b) Similar appearances

Figure 5.12: Robustness evaluation

synthesizing two groups of datasets from the highway dataset: (i) the first group of datasets contains 11 datasets, where the ratio of records with wrongly recognized plates varies from 0% to 100%. Figure 5.12a shows a slight decrease in the accuracy of VeTrac as the ratio increases, while accuracy of LPR drops steeply to 0 when the ratio is large than 50%. (ii) the second group of datasets contains 11 datasets, where the ratio of records with similar appearances varies from 0% to 100%. Figure 5.12b shows a slight decrease in the accuracy of VeTrac as the ratio increases, while accuracy of LPR drops steeply to 0 when the ratio is large than 40%.




(a) Ground Truth of three bus lines (b) Reconstructed trajectories of one vehicle

Figure 5.13: Urban: comparison of reconstructed paths

### Urban

Figure 5.13a illustrates the ground truth of the evaluated three bus lines, where different bus routes are denoted in different colors. The majority of the three bus routes cover different road segments of the area.

Figure 5.13a shows the reconstructed trajectories of one vehicle by VeTrac as well as other baselines. The underlying ground truth of this vehicle is bus line 1. As shown, VeTrac successfully reconstructs the correct trajectory. For the trajectory reconstructed by LPR (denoted in black), it deviates from the ground truth in several parts of the trajectory. This is because LPR can only retrieve snapshots whose license plates are correctly recognized. Errors in LPR texts thus limit the granularity of reconstructed trajectories. Although shortest paths regarding the road networks are introduced to fill the gap between consecutive snapshots, they usually deviates from the group truth and results in incorrect parts of the trajectory. For the trajectory reconstructed by Re-ID (denoted in green), it suffers from a significant deviation from the ground truth. Although more snapshots are recalled by this method, many of the recalled snapshots are the false-positives. Those false-positives are usually snapshots of other vehicles that are similar with the query. The inability of distinguish false-positives limits the precision of reconstructed trajectories and results in long and incorrect trajectories. For the trajectory reconstructed by HRIS (denoted in blue), it is a completely different path comparing with the ground truth. The reason is that the route of this vehicle is different from the most popular route choice. Most of the vehicles sharing the same origin and destination choose a path consists of major roads as denoted in blue.

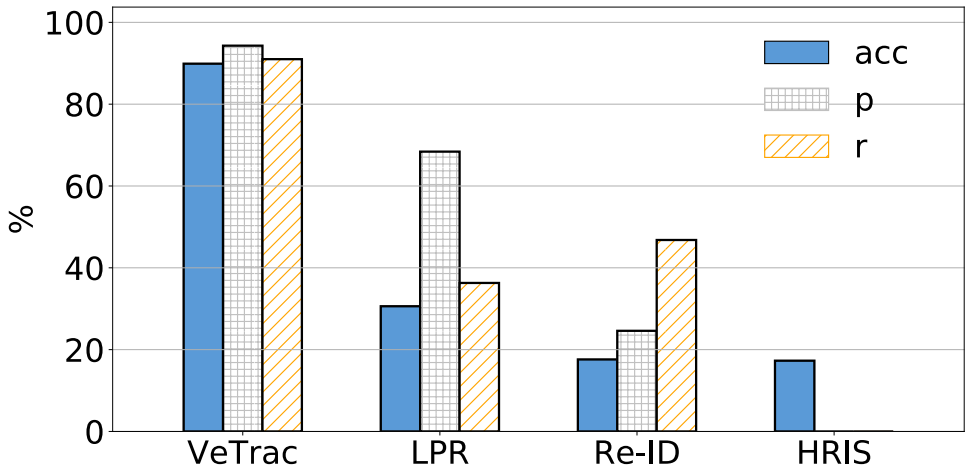


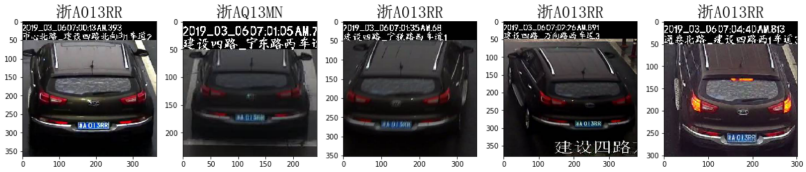

Figure 5.14: Urban: statistical evaluation

We further present the statistical results of reconstructed trajectories. Figure 5.14 shows the average accuracy, precision and recall of VeTrac for both public transits and private vehicles. Overall, the proposed system VeTrac is able to correctly reconstruct 89% of the trajectories, which outperforms other baseline methods by at least 59%. For precision and recall, VeTrac yields trajectories with 94.3% precision and 91.0% recall, which outperforms all other baselines by a large margin. For LPR, the accuracy (30.6%) is limited due to the errors in LPR texts. Compared with statistics in highway scenario(as shown in Figure 5.11), the precision of LPR drops to 68.5% because more trajectories have similar plates in a database of real-world scale. This also affects the Re-ID as more snapshots are visually similar. As a results, the precision of Re-ID drops to 36.3% and the recall of Re-ID drops to 46.5%. For HRIS, since it only reflects the most popular route choice for a given pair of origin and destination, only 17.3% of the trajectories can be correctly reconstructed.

### 5.3.3 Component Performance

We show the effectiveness of proposed components by experiments on the highway dataset.

**Multi-dimensional Similarity Block.** Figure 5.15 shows the effectiveness of proposed MDS with a representative trajectory of five snapshots. As shown, none of the three information dimension (i.e., plate, appearance and spatio-temporally information) can solely rank the five snapshots of a trajectory to the top. However, by fusing infor-




| | | | | | |
|---|---|---|---|---|---|
| **Plate** | 1 | 141 | 1 | 1 | 1 |
| **App.** | 1 | 3140 | 1629 | 343 | 326 |
| **ST** | 2340 | 42 | 24 | 271 | 619 |
| **MDS** | **2** | **5** | **1** | **3** | **4** |

Figure 5.15: The effectiveness of MDS block

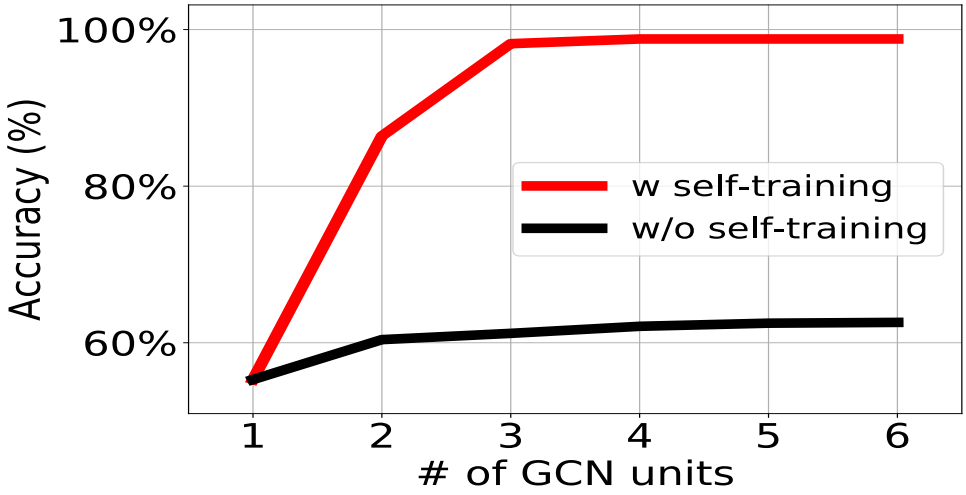



Figure 5.16: The effectiveness of self-training

mation from all dimensions, the proposed MDS block ranks the five snapshots correctly.

**Self-training.** Figure 5.16 shows the effectiveness of proposed self-training. As shown, without the self-training, the accuracy of baseline GCN improves slowly. After reaching 62%, the accuracy stops improving even if more graph convolution units are added. This is because stacking graph convolution units can only help to address incompleted clusters. However, when we have enough depth of graph convolution units, the imprecise clusters are the major problem. The proposed self-training process can handle imprecise clusters and help to improve the accuracy quickly to 98%.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

This thesis studies mobility sensing and analysis in the urban environment. With a special focus on individual-level, community-level and city-level mobility, three pieces of connected studies are presented.

The first work considers the mobility of individual smartphone user. Although various algorithms have been proposed to infer a user's location from data sensed by the smartphone, none of existing localization algorithms can works accurately in all environment, which results in incomplete or noisy mobility. To address this problem, we propose UniLoc, a unified framework that exploits the diversity of existing localization schemes to provide accurate positioning service on mobile devices. We observe that different localization schemes complement with each other at different locations. It is caused by the natures of physical sensors (e.g., error accumulation of gyroscope) or variation of environmental conditions (variant Wi-Fi AP density in different places). The UniLoc framework performs easy aggregation of the end outputs from multiple localization schemes. Our experience suggests that such a simple aggregation can gain much performance improvement. The design philosophy of UniLoc is different from those localization schemes that exploit deep data fusion (e.g., across Wi-Fi, inertial sensors, other landmark sensing, etc.). While the latter involves organic integration of different data sources and rationales of localization, and can be highly optimal with certain environmental conditions (e.g., indoor), UniLoc gives another design dimension to harness the scheme diversity.

For its simplicity, the framework can be easily extended to aggregate the most up-to-date schemes without full knowledge of their design details.

The second study consider the mobility of a community. Aiming at improving the commuting efficiency in urban environment, the study investigating ways to collectively sensing and analyzing the mobility of thousands of students. We propose to use wearable devices as a crowdsensing platform and with that collect massive daily trajectories of students. By carefully processing a large dataset composed of daily trajectories of thousands of students in Singapore, we find that, instead of simply picking up students from their homes, an optimal school shuttle planning system needs to learn the real transportation usage and plan across all potential pickup locations for every student to generate need-satisfying routes. Accordingly, a trip profiling scheme is proposed to learn real transportation usage and extract multiple potential pickup locations for every student, which offers an extra dimension of optimization and more need-satisfying bus plans can thus be derived. To handle extremely large search space brought by massive potential pickup locations, a graph-based data structure is developed to aggregate similar demands with the awareness of road networks; Finally, a Tabu-based search algorithm is designed to efficiently plan the routes that are able to balance the commute time saved for all students and the operating cost of the shuttle buses. Our experiments show that proposed system is able to plan routes that are more beneficial and low-cost than existing solutions.

The third study consider the mobility of all general vehicle in a city. We notice that, although having the moving trajectories of general vehicles in a city serves the foundation in understanding the urban mobility, and with that benefit a board ranges of urban applications, e.g. traffic simulation, congestion reasoning, driver behavior analysis and urban planning. While almost all those applications expect a set of trajectories that are both fine-grained and comprehensive, few works exploited practical methods to profile all the vehicles in a city.almost all existing approaches for vehicle sensing (e.g., on-board device based solutions, roadway sensors based solutions) as they can only acquire either partial or anonymized observations. To address those drawbacks, we propose to utilize widely deployed traffic cameras as a sensor network and design a system to infer vehicle trajectories in road networks based on multiple sources of information. We build similarities on LPR texts, vehicle appearances, and mobility causality between vehicles

snapshots taken from different traffic cameras, and gain combined confidence on their identify correlation. We further propose a graph convolution based model to classify all vehicle snapshots and group them into vehicles of corresponding identities. In order to incorporate the complex mobility dependency among the urban road network, we join the snapshot graph with the road network graph to build a self-learning framework, and improve the accuracy of identities and trajectories profiled for all the vehicles. Our experiments show that proposed system is able to reconstruct trajectories with 89% accuracy.

## 6.2 Future Work

Based on our studies, we point out a few directions as future work that could be valuable to extend our ability of mobility sensing and analysis in urban environment.

**Online error modeling.** Our experience with the experiments tells us that UniLoc (or the way of aggregating different localization schemes) can be further improved by: 1) The error model can be designed more accurate, using more sophisticated regression models, and feature engineering to identify more accurate influence factors [23]; 2) The model training can be extended to online so the localization process itself can serve as an unsupervised training stage to enhance the performance, which will help adapting the error model better to the common environmental conditions of individuals.

**Trajectory data mining.** Trajectories offer fine-grained information for us to better mobility dynamics of individuals, communities, as well as the entire city. As the accumulation of both human and vehicle trajectories by proposed mobility sensing techniques, special trajectory mining algorithms are needed to convert those big datasets into actionable knowledge. By mining the trajectories, we believe insightful knowledge can be obtained for a wide range of applications in transportation and urban redevelopment, e.g., usage-based electronic toll collection (ETC), traffic estimation and prediction, intelligent traffic light control, origin-destination (OD) analysis, planning of functional regions within the city, evaluation and redesign of existing public transportation.

**Fusing of heterogeneous information from multiple sources.** Heterogeneous datasets offer complementary information and help to reveal the facts. When mining mobility and transportation data, it is commonly that data are coming from different

sensors in different forms. For example, we may be visual data and trajectory data to determine the vehicle identity. Thus, the ability of fusing heterogeneous data to achieve better accuracy or optimality is crucial.

**Learning from non-euclidean data.** On the one hand, machine learning and deep learning algorithms are mainly designed for learning from data of euclidean structure. However, a lot of mobility and transportation data are naturally organized in non-euclidean structure, e.g., the transportation network is a graph. Traditional approaches cope with non-euclidean data by transforming them into flat vectors, which however drops important topological information and the achieved results may heavily depend on the transformation. Therefore, the ability of learning from non-euclidean data with the power of advanced machine learning and deep learning is important. On the other hand, the increasing scale of graphs and growing complexity of graph analysis tasks challenge the way of computing. Although we have powerful algorithms and systems to process large euclidean data, parallel and distributed computing on graphs still contains many open questions. We believe it is a crucial problem to solve in order to deploy complex graph analysis applications in real world.

# References


[1] indoo.rs. http://indoo.rs/. Accessed: 2015-12-15.

[2] P. Bahl and V. N. Padmanabhan. RADAR: An in-building rf-based user location and tracking system. In *Proceedings of the International Conference on Computer Communications*, pages 775–784. IEEE, 2000.

[3] B. M. Baker and M. Ayechew. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800, 2003.

[4] B. Balcik, B. M. Beamon, and K. Smilowitz. Last mile distribution in humanitarian relief. *Journal of Intelligent Transportation Systems*, 12(2):51–63, 2008.

[5] P. Banerjee, S. Ranu, and S. Raghavan. Inferring uncertain trajectories from partial observations. In *Proceedings of the International Conference on Data Mining*, pages 30–39. IEEE, 2014.

[6] A. Behboodi, N. Wirstrom, F. Lemic, T. Voigt, and A. Wolisz. Interference effect on localization solutions: signal feature perspective. In *Proceedings of the Vehicular Technology Conference*, pages 1–7. IEEE, 2015.

[7] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

[8] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 245–250, 2001.

[9] R. Bowerman, B. Hall, and P. Calamai. A multi-objective optimization approach to urban school bus routing: Formulation and solution method. *Transportation Research Part A: Policy and Practice*, 29(2):107–123, 1995.

[10] S. T. Buckland, K. P. Burnham, and N. H. Augustin. Model selection: an integral part of inference. *Biometrics*, pages 603–618, 1997.

[11] R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 160–172. Springer, 2013.

[12] C. Cao, Z. Liu, M. Li, W. Wang, and Z. Qin. Walkway discovery from large scale crowdsensing. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 13–24. IEEE, 2018.

[13] P. S. Castro, D. Zhang, and S. Li. Urban traffic modelling and prediction using large scale taxi gps traces. In *Proceedings of the International Conference on Pervasive Computing*, pages 57–72. Springer, 2012.

[14] C. Chen, D. Zhang, N. Li, and Z.-H. Zhou. B-planner: Planning bidirectional night bus routes using large-scale taxi gps traces. *Transactions on Intelligent Transportation Systems*, 15(4):1451–1465, 2014.

[15] H. Chen, F. Li, and Y. Wang. Soundmark: Accurate indoor localization via peer-assisted dead reckoning. *Internet of Things Journal*, 2018.

[16] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pages 233–245. ACM, 2005.

[17] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. Indoor localization without the pain. In *Proceedings of the International Conference on Mobile Computing and Networking*, pages 173–184. ACM, 2010.

[18] S. P. Chuah, H. Wu, Y. Lu, L. Yu, and S. Bressan. Bus routes design and optimization via taxi data analytics. In *Proceedings of the International on Conference on Information and Knowledge Management*, pages 2417–2420. ACM, 2016.

[19] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[20] I. Constandache, R. R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *Proceedings of the International Conference on Computer Communications*, pages 1–9. IEEE, 2010.

[21] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153(1):29–46, 2007.

[22] D. Dearman, A. Varshavsky, E. De Lara, and K. N. Truong. An exploration of location error estimation. In *Proceedings of the Conference on Ubiquitous Computing*, pages 181–198. ACM, 2007.

[23] P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[24] J. Dong, Y. Xiao, M. Noreikis, Z. Ou, and A. Ylä-Jääski. iMoon: Using smartphones for image-based indoor navigation. In *Proceedings of the Conference on Embedded Networked Sensor Systems*, pages 85–97. ACM, 2015.

[25] O. B. Downs, A. Barker, R. C. Cahn, C. H. Chapman, and W. Stoppler. Detecting anomalous road traffic conditions, Mar. 1 2011. US Patent 7,899,611.

[26] N. M. Drawil, H. M. Amar, and O. A. Basir. Gps localization accuracy classification: A context-based approach. *Transactions on Intelligent Transportation Systems*, 14(1):262–273, 2013.

[27] W. Du and M. Li. Harnessing mobile multiple access efficiency with location input. In *Proceedings of the International Conference on Distributed Computing Systems*, pages 246–255. IEEE, 2013.

[28] W. Du, M. Li, and J. Lei. CO-MAP: improving mobile multiple access efficiency with location input. *Transactions on Wireless Communications*, 13(12):6643–6654, 2014.

[29] W. Du, P. Tongx, and M. Lix. Uniloc: A unified mobile localization framework exploiting scheme diversity. In *Proceedings of the International Conference on Distributed Computing Systems*, pages 818–829. IEEE, 2018.

[30] M. E. El Najjar and P. Bonnifait. A road-matching method for precise vehicle localization using belief theory and kalman filtering. *Autonomous Robots*, 19(2):173–191, 2005.

[31] W. A. Ellegood, S. Solomon, J. North, and J. F. Campbell. School bus routing problem: Contemporary trends and research directions. *Omega*, 2019.

[32] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231. ACM, 1996.

[33] D. L. et al. A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 178–189. ACM/IEEE, 2015.

[34] Z. Fang, Y. Yang, S. Wang, B. Fu, Z. Song, F. Zhang, and D. Zhang. Mac: Measuring the impacts of anomalies on travel time of multiple transportation systems. *Proceedings of the conference on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(2):1–24, 2019.

[35] Y. Fu, H. Xiong, Y. Ge, Z. Yao, Y. Zheng, and Z.-H. Zhou. Exploiting geographic dependencies for real estate appraisal: a mutual perspective of ranking and clustering. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 1047–1056. ACM, 2014.

[36] M. Galdi and P. Thebpanya. Optimizing school bus stop placement in howard county, maryland: A gis-based heuristic approach. In *Geospatial Research: Concepts, Methodologies, Tools, and Applications*, pages 1660–1676. IGI Global, 2016.

[37] J. Gao, W. Fan, J. Jiang, and J. Han. Knowledge transfer via multiple model local structure mapping. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 283–291. ACM, 2008.

[38] Gaode Maps. `https://lbs.amap.com/getting-started/path`. Accessed: 2019-05-31.

[39] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt. Micro-blog: sharing and querying content through mobile phones and social participation. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pages 174–186. ACM, 2008.

[40] B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974.

[41] F. Glover. Tabu search. *ORSA Journal on computing*, 1(3):190–206, 1989.

[42] B. L. Golden, S. Raghavan, and E. A. Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[43] Google Directions API. `Developers.google.com/maps/documentation/directions`. Accessed: 2018-01-16.

[44] V. Guihaire and J.-K. Hao. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, 42(10):1251–1273, 2008.

[45] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *Transactions on signal processing*, 50(2):425–437, 2002.

[46] Y. Gwon, R. Jain, and T. Kawahara. Robust indoor location estimation of stationary and mobile users. In *Proceedings of the International Conference on Computer Communications*, pages 1032–1043. IEEE, 2004.

[47] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016.

[48] S. He, S.-H. G. Chan, L. Yu, and N. Liu. Calibration-free fusion of step counter and wireless fingerprints for indoor localization. In *Proceedings of the Conference on Ubiquitous Computing*, pages 897–908. ACM, 2015.

[49] S. He, S.-H. G. Chan, L. Yu, and N. Liu. Maxlifd: Joint maximum likelihood localization fusing fingerprints and mutual distances. *Transactions on Mobile Computing*, 2018.

[50] S. He, S.-H. G. Chan, L. Yu, and N. Liu. SLAC: Calibration-free pedometer-fingerprint fusion for indoor localization. *Transactions on Mobile Computing*, 17(5):1176–1189, 2018.

[51] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. Bayesian model averaging: a tutorial. *Statistical science*, pages 382–401, 1999.

[52] T. Huang and S. Russell. Object identification in a bayesian context. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 97, pages 1276–1282, 1997.

[53] T. Hunter, P. Abbeel, and A. Bayen. The path inference filter: model-based low-latency map matching of probe vehicle data. *Transactions on Intelligent Transportation Systems*, 15(2):507–529, 2014.

[54] O. Javed, K. Shafique, Z. Rasheed, and M. Shah. Modeling inter-camera space–time and appearance relationships for tracking across non-overlapping views. *Computer Vision and Image Understanding*, 109(2):146–162, 2008.

[55] Y. Jin, W.-S. Soh, and W.-C. Wong. Error analysis for fingerprint-based localization. *Communications Letters*, 14(5):393–395, 2010.

[56] V. Kettnaker and R. Zabih. Bayesian multi-camera surveillance. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 253–259. IEEE, 1999.

[57] X. Kong, M. Li, J. Li, K. Tian, X. Hu, and F. Xia. Copfun: An urban co-occurrence pattern mining scheme based on regional function discovery. *World Wide Web*, 22(3):1029–1054, 2019.

[58] X. Kong, X. Song, F. Xia, H. Guo, J. Wang, and A. Tolba. Lotad: Long-term traffic anomaly detection based on crowdsourced bus trajectory data. *World Wide Web*, 21(3):825–847, 2018.

[59] X. Kong, Z. Xu, G. Shen, J. Wang, Q. Yang, and B. Zhang. Urban traffic congestion estimation and prediction based on floating car trajectory data. *Future Generation Computer Systems*, 61:97–107, 2016.

[60] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti. SpotFi: Decimeter level localization using wifi. In *Proceedings of the Conference on Special Interest Group on Data Communication*, pages 269–282. ACM, 2015.

[61] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 59(3):345–358, 1992.

[62] Last Mile. `en.wikipedia.org/wiki/Last_mile_(transportation)`. Accessed: 2017-07-31.

[63] W. Lerner, v. F. Audenhove, et al. The future of urban mobility: Towards networked, multimodal cities in 2050. *Public Transport International*, (2), 2012.

[64] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. A reliable and accurate indoor localization method using phone inertial sensors. In *Proceedings of the Conference on Ubiquitous Computing*, pages 421–430. ACM, 2012.

[65] L. Li, G. Shen, C. Zhao, T. Moscibroda, J.-H. Lin, and F. Zhao. Experiencing and handling the diversity in data density and environmental locality in an indoor positioning service. In *Proceedings of the International Conference on Mobile Computing and Networking*, pages 459–470. ACM, 2014.

[66] Y. Li, Y. Li, D. Gunopulos, and L. Guibas. Knowledge-based trajectory completion from sparse gps samples. In *Proceedings of the International Conference on Advances in Geographic Information Systems*, pages 1–10. ACM, 2016.

[67] Y. Li, J. Luo, C.-Y. Chow, K.-L. Chan, Y. Ding, and F. Zhang. Growing the charging station network for electric vehicles with trajectory data analytics. In *Proceedings of the International Conference on Data Engineering*, pages 1376–1387. IEEE, 2015.

[68] Y. Li, Y. Zheng, and Q. Yang. Dynamic bike reposition: A spatio-temporal reinforcement learning approach. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 1724–1733. ACM, 2018.

[69] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pages 285–298. ACM, 2010.

[70] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye. Push the limit of wifi based localization for smartphones. In *Proceedings of the International Conference on Mobile Computing and Networking*, pages 305–316. ACM, 2012.

[71] K. Liu, X. Liu, and X. Li. Guoguo: Enabling fine-grained smartphone localization via acoustic anchors. *Transactions on Mobile Computing*, 15(5):1144–1156, 2016.

[72] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on Computer Vision*, pages 21–37. Springer, 2016.

[73] X. Liu, W. Liu, H. Ma, and H. Fu. Large-scale vehicle re-identification in urban surveillance videos. In *Proceedings of the International Conference on Multimedia and Expo*, pages 1–6. IEEE, 2016.

[74] X. Liu, W. Liu, T. Mei, and H. Ma. A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In *European Conference on Computer Vision*, pages 869–884. Springer, 2016.

[75] Y. Liu, C. Liu, N. J. Yuan, L. Duan, Y. Fu, H. Xiong, S. Xu, and J. Wu. Intelligent bus routing with heterogeneous human mobility patterns. *Knowledge and Information Systems*, 50(2):383–415, 2017.

[76] Y. Liu, Z. Yang, X. Wang, and L. Jian. Location, localization, and localizability. *Journal of Computer Science and Technology*, 25(2):274–297, 2010.

[77] Z. Liu, Z. Li, M. Li, W. Xing, and D. Lu. Mining road network correlation for traffic estimation via compressive sensing. *Transactions on Intelligent Transportation Systems*, 17(7):1880–1893, 2016.

[78] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the International Conference on Advances in Geographic Information Systems*, pages 352–361. ACM, 2009.

[79] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[80] D. Makris, T. Ellis, and J. Black. Bridging the gaps between cameras. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–II. IEEE, 2004.

[81] B. C. Matei, H. S. Sawhney, and S. Samarasekera. Vehicle tracking across nonoverlapping cameras using joint kinematic and appearance features. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3465–3472. IEEE, 2011.

[82] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2015.

[83] R. Nandakumar, K. K. Chintalapudi, and V. N. Padmanabhan. Centaur: locating devices in an office environment. In *Proceedings of the International Conference on Mobile Computing and Networking*, pages 281–292. ACM, 2012.

[84] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the International Conference on Advances in Geographic Information Systems*, pages 336–343. ACM, 2009.

[85] NSE project. `www.nse.sg`. Accessed: 2017-07-31.

[86] OSM. `www.openstreetmap.org`. Accessed: 2018-01-23.

[87] I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research*, 41(4):421–451, 1993.

[88] V. Otsason, A. Varshavsky, A. LaMarca, and E. De Lara. Accurate GSM indoor localization. In *Proceedings of the Conference on Ubiquitous Computing*, pages 141–158. ACM, 2005.

[89] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pages 299–314. ACM, 2010.

[90] J. Paek, K.-H. Kim, J. P. Singh, and R. Govindan. Energy-efficient positioning for smartphones using cell-id sequence matching. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pages 293–306. ACM, 2011.

[91] S. Papaioannou, H. Wen, Z. Xiao, A. Markham, and N. Trigoni. Accurate positioning via cross-modality training. In *Proceedings of the Conference on Embedded Networked Sensor Systems*, pages 239–251. ACM, 2015.

[92] J. Park and B.-I. Kim. The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311–319, 2010.

[93] J.-g. Park, D. Curtis, S. Teller, and J. Ledlie. Implications of device diversity for organic localization. In *Proceedings of the International Conference on Computer Communications*, pages 3182–3190. IEEE, 2011.

[94] F. Pinelli, R. Nair, F. Calabrese, M. Berlingerio, G. Di Lorenzo, and M. L. Sbodio. Data-driven transit network design from mobile phone trajectories. *Transactions on Intelligent Transportation Systems*, 17(6):1724–1733, 2016.

[95] G. Qiu, R. Song, S. He, W. Xu, and M. Jiang. Clustering passenger trip data for the potential passenger investigation and line design of customized commuter bus. *Transactions on Intelligent Transportation Systems*, 2018.

[96] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: zero-effort crowdsourcing for indoor localization. In *Proceedings of the International Conference on Mobile Computing and Networking*, pages 293–304. ACM, 2012.

[97] U. Raza, A. L. Murphy, and G. P. Picco. Embracing localization inaccuracy: a case study. In *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 207–212. IEEE, 2013.

[98] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[99] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh. Using mobile phone barometer for low-power transportation context detection. In *Proceedings of the Conference on Embedded Network Sensor Systems*. ACM, 2014.

[100] P. Schittekat, J. Kinable, K. Sörensen, M. Sevaux, F. Spieksma, and J. Springael. A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research*, 229(2):518–528, 2013.

[101] SENSg. `www.nse.sg/sensg/about-sensg`. Accessed: 2017-07-31.

[102] Y. Shen, T. Xiao, H. Li, S. Yi, and X. Wang. Learning deep neural networks for vehicle re-id with visual-spatio-temporal path proposals. In *Proceedings of the International Conference on Computer Vision*, pages 1900–1909. IEEE, 2017.

[103] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304, 2016.

[104] Y. Shu, K. G. Shin, T. He, and J. Chen. Last-mile navigation using smartphones. In *Proceedings of the International Conference on Mobile Computing and Networking*, pages 512–524. ACM, 2015.

[105] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[106] Singtel. `www.singtel.com`. Accessed: 2017-07-31.

[107] Skyhook Wireless. `www.skyhookwireless.com`. Accessed: 2017-07-31.

[108] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu. Transportation mode detection using mobile phones and gis information. In *Proceedings of the International Conference on Advances in Geographic Information Systems*, pages 54–63. ACM, 2011.

[109] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik. Indoor localization without infrastructure using the acoustic background spectrum. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pages 155–168. ACM, 2011.

[110] P. Tong, W. Du, M. Li, J. Huang, W. Wang, and Z. Qin. Last-mile school shuttle planning with crowdsensed student trajectories. *Transactions on Intelligent Transportation Systems*, 2019.

[111] C.-C. Tsai. A localization system of a mobile robot by fusing dead-reckoning and ultrasonic measurements. In *Transactions on Instrumentation and Measurement*, volume 1, pages 144–149. IEEE, 1998.

[112] G. Tsoukaneri, G. Theodorakopoulos, H. Leather, and M. K. Marina. On the inference of user paths from anonymized mobility data. In *European Symposium on Security and Privacy*, pages 199–213. IEEE, 2016.

[113] F.-J. Van Audenhove, O. Korniichuk, L. Dauby, and J. Pourbaix. The future of urban mobility 2.0: Imperatives to shape extended mobility ecosystems of tomorrow. 2014.

[114] H. Wang, D. Kifer, C. Graif, and Z. Li. Crime rate inference with big data. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 635–644. ACM, 2016.

[115] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No need to war-drive: unsupervised indoor localization. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pages 197–210. ACM, 2012.

[116] S. Wang, H. Wen, R. Clark, and N. Trigoni. Keyframe based large-scale indoor localisation using geomagnetic field and motion pattern. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1910–1917. IEEE/RSJ, 2016.

[117] Y. Wang, Y. Zheng, and Y. Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 25–34. ACM, 2014.

[118] Z. Wang, L. Tang, X. Liu, Z. Yao, S. Yi, J. Shao, J. Yan, S. Wang, H. Li, and X. Wang. Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification. In *Proceedings of the International Conference on Computer Vision*, pages 379–387. IEEE, 2017.

[119] Wireless@SG. `www.imda.gov.sg/wireless-sg`. Accessed: 2017-07-31.

[120] N. Wirström, A. Behboodi, F. Lemic, T. Voigt, and A. Wolisz. Localization using anonymous measurements. In *Proceedings of the International Conference on Distributed Computing in Sensor Systems*, pages 137–146. IEEE, 2015.

[121] C.-W. Wu, C.-T. Liu, C.-E. Chiang, W.-C. Tu, and S.-Y. Chien. Vehicle re-identification with the space-time prior. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops*, pages 121–128. IEEE, 2018.

[122] L. Wu, B. Yang, and P. Jing. Travel mode detection based on gps raw data collected by smartphones: a systematic review of the existing methodologies. *Information*, 7(4):67, 2016.

[123] Z. Xiao, H. Wen, A. Markham, and N. Trigoni. Lightweight map matching for indoor localisation using conditional random fields. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 131–142. ACM/IEEE, 2014.

[124] J. Xiong and K. Jamieson. ArrayTrack: a fine-grained indoor location system. In *Proceedings of the Symposium on Networked Systems Design and Implementation*, pages 71–84. USENIX, 2013.

[125] C. Xu, B. Firner, Y. Zhang, and R. E. Howard. The case for efficient and robust rf-based device-free localization. *Transactions on Mobile Computing*, 15(9):2362–2375, 2016.

[126] N. Yang and P. S. Yu. Efficient hidden trajectory reconstruction from sparse data. In *Proceedings of the International on Conference on Information and Knowledge Management*, pages 821–830. ACM, 2016.

[127] Z. Yang, C. Wu, and Y. Liu. Locating in fingerprint space: wireless indoor localization with little human intervention. In *Proceedings of the International Conference on Mobile Computing and Networking*, pages 269–280. ACM, 2012.

[128] M. Youssef and A. Agrawala. The Horus WLAN location determination system. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pages 205–218. ACM, 2005.

[129] N. J. Yuan, Y. Zheng, X. Xie, Y. Wang, K. Zheng, and H. Xiong. Discovering urban functional zones using latent activity trajectories. *Transactions on Knowledge and Data Engineering*, 27(3):712–725, 2014.

[130] D. Zhang, Y. Li, F. Zhang, M. Lu, Y. Liu, and T. He. coride: carpool service with a win-win fare model for large-scale taxicab networks. In *Proceedings of the Conference on Embedded Networked Sensor Systems*, pages 1–14. ACM, 2013.

[131] D. Zhang, J. Zhao, F. Zhang, and T. He. UrbanCPS: a cyber-physical system based on multi-source big infrastructure data for heterogeneous model integration. In *Proceedings of the International Conference on Cyber-Physical Systems*, pages 238–247. ACM/IEEE, 2015.

[132] D. Zhang, J. Zhao, F. Zhang, R. Jiang, and T. He. Feeder: supporting last-mile transit with extreme-scale urban infrastructure data. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 226–237. ACM/IEEE, 2015.

[133] H. Zhang, W. Du, M. Li, K. Wu, and P. Mohapatra. Strlight: An imperceptible visible light communication system with string lights. *Transactions on Mobile Computing*, 18(7):1674–1687, 2018.

[134] H. Zhang, W. Du, P. Zhou, M. Li, and P. Mohapatra. An acoustic-based encounter profiling system. *Transactions on Mobile Computing*, 17(8):1750–1763, 2017.

[135] P. Zhang, Z. Bao, Y. Li, G. Li, Y. Zhang, and Z. Peng. Trajectory-driven influential billboard placement. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 2748–2757. ACM, 2018.

[136] W. Zhang, S. Li, and G. Pan. Mining the semantics of origin-destination flows using taxi traces. In *Proceedings of the Conference on Ubiquitous Computing*, pages 943–949. ACM, 2012.

[137] Y. Zhang, W. Hu, W. Xu, H. Wen, and C. T. Chou. Naviglass: Indoor localisation using smart glasses. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks*, pages 205–216, 2016.

[138] K. Zheng, Y. Zheng, X. Xie, and X. Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *Proceedings of the International Conference on Data Engineering*, pages 1144–1155. IEEE, 2012.

[139] Y. Zheng, F. Liu, and H.-P. Hsieh. U-air: When urban air quality inference meets big data. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 1436–1444. ACM, 2013.

[140] Y. Zheng, T. Liu, Y. Wang, Y. Zhu, Y. Liu, and E. Chang. Diagnosing new york city's noises with ubiquitous data. In *Proceedings of the Joint Conference on Pervasive and Ubiquitous Computing*, pages 715–725. ACM, 2014.

[141] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao. Travi-Navi: Self-deployable indoor navigation system. In *Proceedings of the International Conference on Mobile Computing and Networking*, pages 471–482. ACM, 2014.

[142] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the International Conference on World Wide Web*, pages 791–800. ACM, 2009.

[143] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen. IODetector: a generic service for indoor outdoor detection. In *Proceedings of the Conference on Embedded Networked Sensor Systems*, pages 113–126. ACM, 2012.

[144] Y. Zhou and L. Shao. Aware attentive multi-view inference for vehicle re-identification. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 6489–6498. IEEE, 2018.

[145] Z. Zhuang, K.-H. Kim, and J. P. Singh. Improving energy efficiency of location sensing on smartphones. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pages 315–330. ACM, 2010.