Human Activity Detection using Smartphones and Maps

BY
LEON O. STENNETH
B.Sc (University of Technology, Jamaica) 2003
M.Sc (University of Westminster, London, UK) 2004

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2013

Chicago, Illinois

Defense Committee:
- Professor Ouri Wolfson, Computer Science, Chair and Advisor
- Professor Philip Yu, Computer Science, Co-advisor
- Professor Prasad Sistla, Computer Science
- Professor Jane Lin, Civil and Materials Engineering
- Dr. Dr. Bo Xu, Argonne National Laboratory

This thesis is dedicated to my family without whom it would have never been accomplished.

ACKNOWLEDGEMENTS

My first debt of gratitude goes to my advisers Professor Ouri Wolfson and Professor Philip Yu. They have provided skillful guidance that helped me accomplish my research goals. They have been very supportive since I commenced the PhD program. I cannot thank them enough for their hard work, fruitful comments, and diligence. Thank you Professor Wolfson and Professor Yu.

I would like to give wholehearted thanks to my examination committee inclusive of Professor Jane Lin, Professor Prasad Sistla, and Dr. Dr. Bo Xu. I am honored to have such dynamic committee who provides significant comments and suggestions that improve the research.

I would also like to thank my family, coworkers, friends, and everyone who supported me through this dissertation. Without their support this dissertation would not exist. Finally, I would like to thank the National Science Foundation (NSF) and Fulbright for their generous funding support.

LS

TABLE OF CONTENTS

1. INTRODUCTION 1
  1.2 Background and Scope 2
  1.3 Technical challenges 4
    1.3.1 Energy and memory constraints on mobile phones: 4
    1.3.2 Transportation mode activity detection: 5
    1.3.3 Parking status detection 5
    1.3.4 Parking availability estimation: 7
    1.3.5 Privacy aware mobile sensing: 7
  1.4 Methodology 8
  1.5 Contributions of work 9
    1.5.1 Outdoor mobile sensing 10
    1.5.2 Indoor and outdoor mobile sensing 11
    1.5.3 Parking status detection 11
    1.5.4 Street parking availability estimation 13
    1.5.5 Privacy-aware mobile sensing 13
  1.6 Applications 14
2. LITERATURE REVIEW 19
  2.2 Activity detection 19
    2.2.1 Commercial devices 19
    2.2.2 Custom hardware 20
    2.2.3 Mobile phones 21
  2.3 Street parking status detection and availability estimation 24
  2.4 Privacy aware mobile sensing 26
    2.4.1 Snapshot queries 26
    2.4.2 Continuous query 29
3. METHODOLOGY 31
  3.2 Summary of general model 31
  3.3 Smartphone sensors 32
    3.3.1 Accelerometer sensor 33
    3.3.2 GPS sensor 33
    3.3.3 Bluetooth Sensor 33
    3.3.4 Smartphone sensor reports 34
  3.4 Geospatial data (transportation network) 35
    3.4.1 Real time bus location 35
    3.4.2 Bus Stops 36
    3.4.3 Rail Lines 36
  3.5 Supervised machine learning model for activity detection 37
  3.6 Street parking 38
4. OUTDOOR ACTIVITY DETECTION 40
  4.2 The GPS and GIS model for outdoor activity detection 41
  4.3 Classification features for outdoor transportation mode sensing 42
    4.3.1 Average accuracy of GPS coordinates 43

TABLE OF CONTENTS (continued)

4.3.2 Average speed 43
4.3.3 Average heading change 44
4.3.4 Average acceleration 44
4.3.5 Bus location closeness 45
4.3.6 Average bus closeness (ABC) 45
4.3.7 Candidate bus closeness (CBC) 46
4.3.8 Rail line trajectory closeness 46
4.3.9 Bus stop closeness rate 47
4.4 The application environment 48
4.5 GPS data preprocessing 50
4.6 Spatial indexing 50
4.6.1 Spatial index performance evaluation 51
4.7 Evaluation of GPS and GIS model 53
4.7.1 Classification without transportation network 53
4.7.2 Classification with Transportation network 55
4.7.3 Extended real world evaluation of the deployed system 57
4.7.4 Discussion 59
4.7.5 Relevant work 60
5. INDOOR AND OUTDOOR ACTIVITY DETECTION 63
5.2 Sensor selection and transportation network 65
5.2.1 Sensors 65
5.2.2 Transportation Network 67
5.3 General algorithm 67
5.4 Classification feature extraction 68
5.5 Architecture for indoor and outdoor transportation mode detection 72
5.6 Mobile device position 73
5.7 Evaluation by experiments 78
5.7.1 Experimental setup and sensor data collection 78
5.7.2 Classifier Selection 80
5.8 Results 82
5.8.1 Indoor transportation mode detection 83
5.8.2 Outdoor transportation mode detection 87
5.8.3 Indoor and outdoor transportation mode detection 89
5.8.4 Mobile phones with GPS or accelerometer turned off 91
5.8.5 Battery power consumption 93
5.9 Limitations of study 95
5.10 Discussion 95
5.11 Relevant work 97
6. PARKING STATUS DETECTION (PSD) 100
6.2 Parking status detection 100
6.2.1 Connection to in-vehicle Bluetooth 101
6.2.2 Transportation mode monitoring 102

TABLE OF CONTENTS (continued)

6.2.3 Pay-by-phone Piggyback 105
6.3 Energy conservation study 106
6.4 Parking and deparking detection accuracy 107
6.4.1 Automated parking detection accuracy 107
6.4.2 Unparking or deparking detection accuracy 108
6.5 Relevant work 111
7. PARKING AVAILABILITY ESTIMATION (PAE) 113
7.2 Overview of parking availability estimation 114
7.3 Constructing Historical Profiles 117
7.3.1 Historical Availability Profile 117
7.3.2 HAP Construction Algorithm 118
7.3.3 Computation of Confidence 119
7.3.4 Assumptions 120
7.4 Parking Availability Estimation algorithms 123
7.4.1 Historical Statistics (HS) 124
7.4.2 Scaled PhonePark (SPP) 124
7.4.3 Weighted Average (WA) 124
7.4.4 Kalman Filter (KF) 125
7.5 Evaluation by Simulations 126
7.6 Evaluation of HAP 129
7.6.1 Tuning-up Weighted-Average (WA) 132
7.7 Comparison of HS, SPP, WA, and KF 133
7.8 Comparison of RMSE 135
7.9 Comparison on Boolean availability accuracy 136
7.10 Relevant work 138
7.11 Discussion 140
8. PRIVACY AWARE MOBILE SENSING 142
8.2 Preliminaries 145
8.2.1 Formal definitions 145
8.3 Requirements 146
8.4 System architecture 148
8.4.1 Mobile device (clients) 148
8.4.2 Anonymization server (AS) 148
8.4.3 Secure communication service 149
8.4.4 Service provider (LBS) 149
8.4.5 Operation flow 149
8.5 Threat model and adversary's knowledge 150
8.5.1 Count (R) 151
8.5.2 Lookup (lat, lon) 151
8.5.3 Intersect (R_1, R_2, R_3, R_4 … R_{n-1}, R_n) 151
8.6 Dummy requests 151
8.6.1 Dummy profile 152

TABLE OF CONTENTS (continued)

8.6.2 Dummy diversity..156
8.7 Query privacy in MobiPriv implies location privacy..157
8.7.1 Proof that query privacy implies location privacy for type 1 queries..158
8.8 MobiPriv: System overview..159
8.8.1 Mobile request..161
8.8.2 Transformation and mapping function..162
8.9 Algorithms..162
8.9.1 CloakLessK..163
8.9.2 CloakedK..164
8.10 Complexity analysis discussion..165
8.11 Proof of correctness for MobiPriv in snapshot queries..166
8.12 Evaluation..167
8.12.1 Evaluation criteria..168
8.13 Experimental setup and road network..170
8.14 Experimental results..172
8.14.1 High privacy requirement and high spatial tolerance..172
8.14.2 Lower privacy requirement and high spatial tolerance..175
8.14.3 Lower privacy requirement and low spatial tolerance..177
8.14.4 Privacy and Quality of Service (spatial tolerance)..181
8.15 Continuous queries..183
8.15.1 Query linking in continuous queries..183
8.15.2 Discussion on K_{global} and K_{local}..185
8.15.3 MobiPriv prevents corollary history attacks..185
8.15.4 Proof of correctness for continuous queries..187
8.15.5 Resilience in continuous queries..188
8.15.6 Resilience across snapshots..189
8.15.7 Resilience and Quality of Service (spatial tolerance)..190
8.15.8 Resilience and global privacy requirement..191
8.16 CloakLessK and CloakedK..192
8.17 Discussion..193
9. CONCLUSION AND FUTURE WORK..195
10. CITED LITERATURE..199
CURICULUM VITAE..206

LIST OF FIGURES

Figure 1-Central server model for activity detection........ 4

Figure 2- Fine granularity of bus mode detection 16

Figure 3- Map of street parking availability........ 17

Figure 4-Map of street parking availability........ 18

Figure 5 – Classification examples........ 38

Figure 6-Components of the PhonePark system........ 39

Figure 7 - GPS/GIS activity detection model........ 42

Figure 9-Spatial index performance 52

Figure 10-Transportation network features not considered........ 54

Figure 12-Deployed classification system........ 57

Figure 14- Classification examples 68

Figure 15- Accelerometer raw data sampled at 50 times/second. Vertical dimension is y. 69

Figure 16- System architecture (centralized model)........ 72

Figure 17- mobile device in bag (x axis readings) 74

Figure 18 - mobile device in bag (y axis readings) 75

Figure 19-mobile device in bag (z axis readings)........ 75

Figure 20-mobile device in front pocket (x axis readings)........ 76

Figure 21-mobile device in front pocket (y axis readings)........ 76

Figure 22-mobile device in front pocket (z axis readings) 77

Figure 23-mobile device on waist (x axis readings)........ 77

Figure 24-mobile device on waist (y axis readings)........ 78

Figure 25 – mobile device on waist (z axis readings) 78

LIST OF FIGURES (continued)

Figure 26-Data distribution 80

Figure 27- Battery power consumption and sensor usage (Apple’s IPhone 3G) 94

Figure 28- Parking detection using Bluetooth connection between phone and car 101

Figure 29-Parking finite state machine 103

Figure 30-Simplified parking/deparking detection based on transportation mode transitions 105

Figure 31-Parking status detection by pay by phone piggyback 106

Figure 31- Parking detection accuracy 107

Figure 32-unparking or deparking detection accuracy 109

Figure 33- PSD generates spatial and temporal parking and deparking reports 110

Figure 34-PSD generates spatial and temporal parking and deparking reports 110

Figure 35-Components of the PhonePark estimation system 115

Figure 36-Street blocks considered for PAE evaluation 127

Figure 37-RMSE of HAP, b=1%, Polk St. 130

Figure 38-RMSE of HAP, b=50%, Polk St. 131

Figure 39-RMSE of HAP, b=1%, Chestnut St. 131

Figure 40-RMSE of HAP, b=50%, Chestnut St. 132

Figure 41-Determining weight (WHS) for WA 133

Figure 42-RMSE of PAE algorithms, b=1%, Polk St. 134

Figure 43-RMSE of PAE algorithms, b=50%, Polk St. 134

Figure 44-RMSE of PAE algorithms, b=1%, Chestnut St. 135

Figure 45 - RMSE of PAE algorithms, b=50%, Chestnut St. 135

Figure 46-Boolean availability accuracy, b=1%, Polk St. 137

LIST OF FIGURES (continued)

Figure 47-Boolean availability accuracy, b=1%, Chestnut St. 137

Figure 48-LBS with Anonymization Server (AS) 150

Figure 49-Dummy request generation 153

Figure 50 (a,b) -MobiPriv Query Privacy implies Location Privacy 158

Figure 51 - Evaluation area (Map of Chamblee Region, Georgia, USA) 171

Figure 52-success rate and number of mobile requests 172

Figure 53- cloaking time and number of mobile requests 173

Figure 54-communication cost and number of 174

Figure 55 - success rate and number mobile 176

Figure 56 - cloaking time and number of mobile requests 176

Figure 57 - communication cost and number mobile 177

Figure 58 - success rate and number of mobile 178

Figure 59-cloaking time and number of mobile 179

Figure 60 - communication and number of mobile requests 180

Figure 61 - success rate and quality of service (spatial tolerance) 181

Figure 62 - cloaking time and spatial tolerance 182

Figure 63 - communication cost and spatial tolerance 183

Figure 64 (a, b, c) - Corollary history attack in continuous queries 184

Figure 65 - resilience across five snapshot queries within a continuous query 189

Figure 66 - resilience and quality of service 190

Figure 67-Resilience and global privacy 192

LIST OF TABLES

Table 1 - sensor data distribution 49

Table 2-Further performance statistics 52

Table 3-Evaluation of deployed system 59

Table 4- Relate work with GPS sensor 62

Table 5- F Score (GPS/GIS and accelerometer model) 83

Table 6-Confusion matrix for accelerometer sensor only 85

Table 7-Confusion matric for GPS/GIS model 86

Table 8- F Score of outdoor transportation mode detection (GPS/GIS and accelerometer) 88

Table 9-F Score of indoor and outdoor transportation mode detection 90

Table 10-Confusion matrix of Decision Tree (DT) 91

Table 11-Average precision when sensors are turned off for indoor and outdoor modes 92

Table 12 - System parameters and their values. 129

Table 13- Road Properties 171

LIST OF ABBREVIATIONS

AS – Anonymization server

b – Penetration ratio of PSD device

BT – Bluetooth

B – Bus transportation mode

C - Car transportation mode

CK- CloakedK location privacy algorithm

DT – Decision Tree

fp – False positive probability

fn – False negative probability

GPS – Global positioning satellite

GSM - Global System for Mobile Communications

HAP – Historical availability profile

K – K in K-anonymity

LBS – Location based system

PSD – Parking status detector

PAE – Parking availability estimator

Py – Casper’s Pyramid privacy scheme

RF – Random Forest

S – Stationary transportation mode

SFPark – San Francisco Park

LIST OF ABBREVIATIONS (continued)

W – Walking transportation mode

T- Train transportation mode

SUMMARY

This dissertation provides systems, methods, and algorithms for the detection of human activities from smartphones and maps. Knowledge of automated human activities derived from travelers' smartphones enables data service innovations where mobile services are created based on patterns related to individuals as well as communities. Some examples of human activities that we propose to derive from travelers' mobile phones are transportation modes (indoor and outdoor) and street parking status. Using information about individual travelers, the dissertation shows how community patterns such as the parking availability on a street block can be estimated.

Automated detection of human activities helps in data collection. In travel demand transportation surveys, travelers are expected to report the types of transportation mode that they utilized for past trips. Some data collection strategies rely on manual labeling of data after the trip. Thus, inaccuracies are introduced. For example, a traveler may not recall the exact time that they boarded a transportation mode. Since data collected by these manual strategies rely on the traveler's recollection after trips were taken, travel times, durations, locations, and distances recorded by these surveys may contain inaccuracies. Using mobile phone sensors such as GPS and accelerometer on travelers' mobile phones is promising for human activity detection and improves human activity sensing procedures. For another scenario, if we discover from Mary's smartphone sensors that she is traveling by car, then she can be sent a gas coupon or a service special automatically.

Apart from manually denoting ones activity, traditional methods for detecting human activities rely on custom hardware such as multiple accelerometers placed at different body positions. We propose a solution that utilizes sensors on the smartphone of the traveler. From these sensors, we will identify and mobility patterns that are unique to each activity to be detected. The proposed method relies on mobility sensor data from the smartphone sensors such as GPS and accelerometer in conjunction with geospatial knowledge of the underlying transportation network. Given these mobility patterns, machine learning strategies for identifying the current or past activities of the traveler is considered. These mobility patterns of the traveler obtain from their smartphone sensors and the transportation network are utilized as training examples for the machine learning model. The machine learning model then identifies the activity of the traveler probabilistically. The proposed method for human activity detection has been applied to outdoor transportation mode activity recognition, indoor transportation mode activity recognition, and street parking status detection.

For outdoor transportation mode detection, the GPS sensor on the traveler's mobile phone and geospatial knowledge of the underlying transportation network is utilized. Geospatial knowledge of the underlying transportation network includes real time bus locations, bus stop locations, and rail line geometry. Outdoor transportation mode includes buses, cars, trains, etc.

For indoor transportation mode detection, in addition to GPS and geospatial data, we considered the mobile phone's accelerometer sensor. Indoor transportation mode includes stairs, escalators, and elevators.

For street parking status detection (PSD), three approaches are proposed. The first utilizes transportation mode activities as inputs to a street parking finite state machine. This machine monitors the sequence of changes in the detected transportation modes of the traveler. The second approach considers the paring of the Bluetooth sensor on the traveler's mobile phone with the in-car Bluetooth component. The third approach listens for pay for parking transactions from the traveler’s mobile phone.

This dissertation also focuses on algorithms for parking availability estimation (PAE) for a given street block. These parking estimation algorithms consider real-time parking observations obtained from sparse parking status detectors aggregated with historic information (i.e. mean and variance) about parking availability on the given street block. The historic information is referred to as historic availability profile and is derived from sparse and error prone parking status detectors.

A realistic concern that arises from the use of location data on a traveler’s mobile phone is the end user’s privacy. Location privacy in mobile systems is addressed in this thesis. Specifically, K-anonymity techniques using fake user generation to confuse adversaries is introduced.

Future work involves the evaluation of other low power sensors such as GSM for activity detection and constructing classification models for parking that take weather, events (e.g. soccer games), and demographic information on a street block into consideration for parking availability estimation. Further work will balance the tradeoffs of sensor privacy preservation (e.g. location privacy) and activity recognition in mobile systems.

1. INTRODUCTION

The mobile industry is witnessing a rise in the number of smartphones with context aware mobile services that collect sensor data from mobile phone users, analyze these data sets to identify higher order attributes, and then feed these abstractions back to serve the users. This gives rise to sensor data driven service innovation, where mobile service ideas are generated based on patterns related to individuals as well as social networks. The key idea is to create compelling user experiences based on the patterns extracted from the sensors on the traveler's mobile phone. Awareness of a user's activities helps to determine the user's carbon footprint [1, 2], or track the amount of calories burnt [3, 4]. These benefits are derived directly from the ability to automatically detect the activities of persons from their mobile phones. For another example, the transportation mode activities, such as walking, cycling, or train denotes some important characteristics of the mobile user's context. With knowledge of a traveler's transportation mode, targeted and customized advertisements may be sent to the traveler's device. For example, if we discover that Alice is driving by car, the system may send her gas coupons or vehicle service specials. For another example, if we discover that Bob is traveling by bus, then his sensor reports cannot be used to predict the expected travel time on a road network. This is important for traffic and navigation providers that use travelers as probes.

This dissertation studies transportation mode detection using different combination of mobile phone sensors (e.g. GPS, accelerometer) and geospatial knowledge of the transportation network. The dissertation shows how we can derive parking activities by using the detected transportation mode of a traveler. For example, a parking activity is the sequence of transportation mode changes driving → stop → walking while a deparking activity is walking → stop → driving.

Apart from using transportation mode changes, parking status detection via Bluetooth is also studied. Parking status detection enables parking and deparking activities to be detected at a single street parking slot level.

Parking availability estimation using parking status detectors and historical parking information is another contribution of the thesis. Parking availability estimation is not trivial, it takes into consideration that the penetration ratio of the parking status detectors may be low and that parking status detectors can have false positive and false negative errors. Thus, each parking or deparking report from a parking status detector is received with some probability.

Further, privacy preserving mobile sensing is also addressed and algorithms are introduced that provides location privacy for end users in a mobile sensing environment.

The rest of this chapter discusses the background, technical challenges, and scope of the research. Chapter 2 presents the literature review. Chapter 3 describes the research methodology that is considered for this research. Chapter 4 presents mobile sensing and its application to outdoor transportation mode activity detection. Chapter 5 discusses indoor and outdoor activity recognition using mobile sensing. Chapter 6 presents parking status detection and Chapter 7 discuses parking availability estimation. Chapter 8 discuses location privacy preservation in mobile systems and the conclusion and further work are in Chapter 9.

1.2 Background and Scope

The scope of the research is mobile sensing. Specifically it focuses on the detection of human activities from their smartphones and maps. Maps include geospatial knowledge of the underlying transportation network such as real time bus spatial and temporal properties, bus stop, rail line, and parking pay box spatial properties. On the traveler’s smartphone, there exist several sensors such as GPS, accelerometer, gyrometer, microphone, camera, and many others. We will utilize these sensors with geospatial knowledge of the transportation network for human activity detection.

Within the context of mobile sensing, there are generally two paradigms for activity recognition. These two paradigms are the central server and distributed model. With the central server model, the sensor reports from the smartphone are submitted over a network to a central authority for processing and activity inference.

In the distributed paradigm, the sensor reports are not submitted to a central authority, instead they are processed locally on the smartphone for activity inference. The classification processing of the traveler’s activity is therefore done directly on the smartphone. Both paradigms have advantages and disadvantages. For example, in the distributed model since geospatial knowledge is considered, this geospatial knowledge is cached on the device and consumes storage resources. In the central server approach, geospatial knowledge is archived on the server where storage space consumption may not be a significant issue (see Figure 1).

In the central server mobile sensing model, there is communication between the server and the mobile device. Consequently, this model consumes more bandwidth. This centralized system is more platform independent than the distributed counterparts, where classification is done locally on the mobile device. This research dissertation focuses on the central server model and the sensor reports from the traveler’s mobile phone are not batched before transmission to the central server. Instead, they are transmitted from the mobile phone immediately when they are available. The mobile phone is accessed periodically using software for sensor reports.

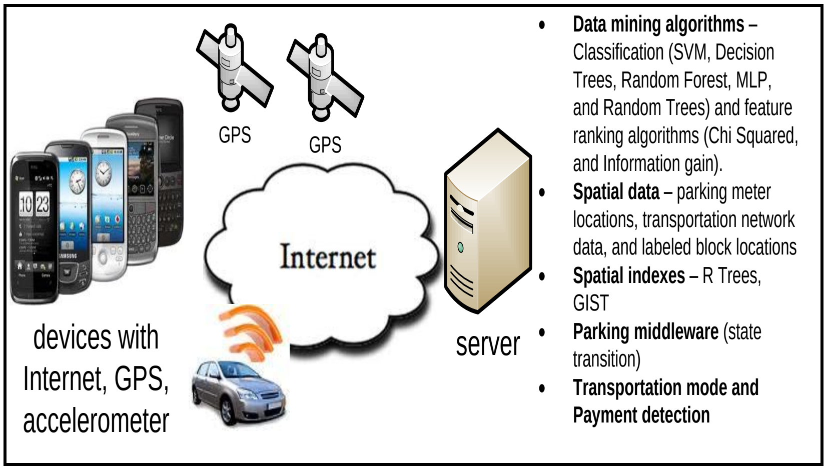

GPS
GPS
Internet
server
devices with Internet, GPS, accelerometer
Data mining algorithms – Classification (SVM, Decision Trees, Random Forest, MLP, and Random Trees) and feature ranking algorithms (Chi Squared, and Information gain).
Spatial data – parking meter locations, transportation network data, and labeled block locations
Spatial indexes – R Trees, GIST
Parking middleware (state transition)
Transportation mode and Payment detection

Figure 1-Central server model for activity detection

This dissertation also studies sensor privacy preservation on the central server model. Specifically, we focus on location privacy which involves ensuring that a mobile user's location is indistinguishable from other user locations.

1.3 Technical challenges

In this section, the technical challenges of the research are highlighted and discussed.

1.3.1 Energy and memory constraints on mobile phones:

Since the sensors on the traveler's mobile phone are accessed for activity recognition and harnessing mobile phone sensors is power consuming. One challenge is to balance the tradeoffs of power consumption and mobile sensing. In other words, the algorithms should be energy-aware. Energy aware strategies such as temporally selectively sampling mobile phone sensors, spatially selectively sensor sampling (e.g. when the traveler is outdoors only), and sensor sampling of low power sensors are studied. Furthermore, since geospatial knowledge (e.g. real time bus locations, road geometries, and bus stop locations) of the underlying transportation network is utilized, memory constraints and indexing for performance purposes (e.g. speed of classification feature creation) establish challenges.

1.3.2 Transportation mode activity detection:

In any mobile sensing environment, travelers utilize various transportation modes. These transportation modes include both outdoor and indoor transportation modes. Indoor transportation modes are for example, stairs, elevator, and escalator. One challenge with indoor transportation mode detection is the fact that sensors such as GPS may be inaccurate indoors since the mobile device does not have a clear view of the positioning satellites while indoors. Hence, the location reported by the GPS sensor on a travelers' mobile phone while indoors is imprecise. In general, GPS is inaccurate and is a technical challenge for outdoor transportation mode detection as well. However, this issue is more noticeable for indoor transportation modes.

For the case of outdoor transportation mode activity detection, apart from the GPS inaccuracy issues, several challenges exist. One challenge is how to distinguish between motorized modes such as cars, buses, and trains since the mobility patterns of motorized modes are similar. For example, cars, buses, and trains can travel at almost the same speed. Likewise, bikes and cars can have the same speed and heading mobility patterns in vehicular traffic congestion. Thus, detecting and distinguishing different motorized transportation modes from smartphone sensor data is not trivial.

Finally, not all mobile phones have the sensors that we propose to utilize for human activity detection. The challenge is develop algorithms that consider the common sensors on mobile phones (e.g. GPS).

1.3.3 Parking status detection

For parking status detection, three schemes are introduced (transportation mode transitions, Bluetooth, and pay by phone piggyback). For the transition scheme, transportation modes detected from sensors (i.e. GPS and accelerometer) are first used as input to a parking finite state machine. Given these activity transition inputs, the parking finite state machine can detect when and where the traveler parked their car. For example, if the mobility transition sequence car → stationary → walking is observed, then the parking status detector (PSD) infers that the driver parked at the stationary point. Several challenges exist for parking status detection. An observation of the PSD may be incorrect due to many reasons. First, the classification of transportation mode may be wrong due to sensor errors and ambiguity between different transportation modes. For example, the mobility pattern of walking is similar to that of driving in a heavy traffic situation. Thus, the system may concur a false positive parking activity in the event a car was driving (drive or car mode), stopped at the stop light (stop mode) and then enters a congested traffic zone (misclassified walk mode). Another challenge is that regardless if the transportation modes are correctly tracked, the parking/deparking inference from the transition sequence may be wrong. For example, the transition sequence car → stationary → walking may be the result of a passenger being dropped (as opposed to a driver parking). Thus, techniques are needed to distinguish a driver from a passenger of a car. Another challenge in parking status detection is GPS accuracy. GPS accuracy is a few meters and gets worst with increase cloud cover, urban areas, under bridges, and canyons. Subsequently, a detected parking or deparking activity may not be at the true location.

Further challenges involve the use of the Bluetooth sensor for parking status detection. The Bluetooth method introduced in this dissertation relies on the pairing of the driver’s Bluetooth on their mobile phone and the in-car Bluetooth component. False positives of parking activities may be observed if a passenger’s mobile phone’s Bluetooth is paired with the in-car Bluetooth. Strategies are proposed to alleviate these challenges to make the system robust and effective.

1.3.4 Parking availability estimation:

Given parking status information of drivers, another challenge is how to estimate the number of slots that are available on a street block in real time. First, not all drivers are equipped with a parking status detection enabled mobile phone. Thus, the received parking observations represent a fraction of the true parking activities and these activities may be sparse throughout the city. We refer to this as the penetration ratio of the parking status detectors. The parking availability estimation algorithms should provide accurate estimation of parking availability under low penetration ratio such as 1%. Apart from the issue of low penetration ratio, each parking status detection event is received with false positive and false negative probabilities. False negative is the probability of a parking or deparking event occurring and was not detected even though the driver possess a parking status detector. False positive is the probability that a parking or deparking event did not occur and the driver's parking status detector reports a parking observation. All the errors of PSD observations propagate to the PAE component when PAE aggregates these observations to estimate the number of available parking spaces. Furthermore, when PAE scales PSD observations to compensate for the market penetration ratio, there are sampling errors. PAE takes all these errors into account when combining the real-time observations with historical statistics.

1.3.5 Privacy aware mobile sensing:

In this dissertation, for activity recognition we rely on GPS among other mobile phone sensors. If these location data acquired from GPS are not protected adequately, they may cause privacy breeches. Several reports are available where people have been stalked through GPS [5]. In this dissertation, the desired level of privacy is personalized and each user can specify the level of location privacy that they desire. Obfuscation is based on the location of other users or requests in the system. The main challenge of privacy aware systems that rely on K-anonymity such as ours is if K-1 other requests or users cannot be found, the location based query is discarded because the desired privacy cannot be provided [6,7]. Consequently, one objective is to minimize the amount of discarded queries. In other words, maximize the amount of queries that can be successfully anonymized.

Another challenge is the balancing of tradeoff between the desired level of privacy by the user and quality of service (QoS) desired. This dissertation shows how to address these challenges to guarantee that all location based request can be processed with the desired level of privacy.

1.4 Methodology

This dissertation adopts a central server paradigm which exploits smartphones to cope with extensive real time updates from the real time tracking of several thousand buses, also spatial properties of several thousand bus stops and parking meters [8]. Specifically, sensor reports from the traveler's mobile phone are submitted to the central server. The smartphones are clients, and clients do not communicate with each other for activity recognition. A sensor report is a set of attribute values sensed (e.g. GPS location and timestamp as obtained from the mobile phone's GPS sensor). Given the sensor reports from mobile phones that are sent to the central server, the central server then executes the proposed algorithms and probabilistically determine the activity of the subject that submitted the sensor reports. Using the central server paradigm memory and storage limitations of the mobile phones are mitigated since the geospatial data can be archived on the central server.

Both activity recognition and parking availability estimation considers the central server scheme. For parking, a PSD runs at a mobile device (client). It detects when and where a driver parks/deparks. When the PSD recognizes a parking or deparking activity, it submits a report to the PAE component indicating that a parking space is occupied or released. The PAE component is on the central server and aggregates the reports of individual PSDs to estimate the number of available parking slots in each street block. For estimation, PAE considers the facts that not all drivers are equipped with the PSD component (i.e., the market penetration ratio of the PSD is not 100%) and the mobile device can have false positive and false negative possibilities for parking detection. Furthermore, to compensate for the inaccuracy of PSD observations, it combines PSD observations with historical parking statistics to get the final estimate. Historical parking statistics is obtained from the proposed historic availability profile (HAP) construction algorithm while PSD observations are obtained from the drivers with the PSD devices.

One consideration with the central server approach is preservation of the traveler's privacy since their sensor reports (e.g. GPS reports) are submitted to a central authority. Location privacy preservation aims to prevent adversaries from learning a mobile user's past or current locations, or the times of the visits. This dissertation addresses the problem of privacy preservation in mobile sensing systems. The privacy aware sensing system consists of mobile devices with positioning capabilities, location based services (LBS), wireless networks, and the proposed privacy preservation algorithms running on a privacy aware middle-ware called an anonymization server (AS) that is located between the clients and the central server. This approach is referred to as privacy-aware mobile sensing [6, 7, 9, 10].

1.5 Contributions of work

In this section, we highlight several contributions of the dissertation. First, the contributions on outdoor activity detection are discussed. Next, contributions on both indoor and outdoor activity recognition are presented. Additionally, the dissertation presents contributions to parking status detection, constructing parking historical availability profiles, and estimating street parking availability. Finally, the thesis discusses contributions in privacy aware mobile sensing.

1.5.1 Outdoor mobile sensing

The dissertation explored the use of the smartphone's GPS sensor and geospatial knowledge of the transportation network to automatically detect the traveler's outdoor transportation mode in real-time [8]. More specifically, using GPS and geospatial data, we developed new supervised machine learning classification features that can effectively distinguish between the modes car, bus, train, bike, walk, and stationary. The propose approach clearly distinguishes between motorized modes and quantifies the fact that including transportation network into the model makes motorized transportation mode improves the detection probability of motorized modes and bikes. Previous work did not distinguish between motorized modes such as cars, buses, and trains with such high accuracy [8, 11, 12, 13, 14, 15]. Distinguishing between motorized and non-motorized transportation mode is not a difficult problem. However, with multiple motorized transportation modes, the problem becomes more difficult since buses, cars and trains may have similar GPS or accelerometer readings. We show that using a transportation network with real time and static spatial data, we can obtain high detection accuracy for various motorized and non-motorized transportation modes. The thesis contributions to outdoor transportation mode detection are as follows: (1) In addition to the traditional features on average speed and average acceleration, we identify for the first time the features of average bus closeness, average rail closeness, and average candidate bus closeness as the most effective features related to transportation mode detection, (2) The proposed work is the first to distinguish between motorized modes (bus, car, train) with such high accuracy, (3) The dissertation introduces a zip-code based indexing and pruning technique to speed up the feature computation. The tradeoff of using such an indexing scheme compared to other spatial indexing techniques such as R-Trees [16] is measured, and (4) Presentation of simulation and real world results, showing the efficiency and effectiveness of the proposed approach.

1.5.2 Indoor and outdoor mobile sensing

In addition to automatically determining outdoor transportation modes such as car, bus, train, walk, and stationary, we propose a classification scheme to detect indoor transportation modes such as elevator, escalator, and stairs. Detecting indoor transportation mode is important for activity recognition. For example, for automatic calorie computations, it makes sense to automatically detect whether the traveler utilizes a stair case or elevator. A person that is using stairs should burn more calories than if they ride the elevator for the same trip. Another motivation for automatic indoor transportation mode detection is determining whether an escalator/elevator is working, or if it exists. There are 2 cases to consider: (1) It is known that stairs/elevator/escalator exists, the question is whether the smartphone is using them, (2) Inference of the existence of stair/elevator/escalator at a location.

Thus, GPS, accelerometer, and transportation network data is combined for indoor and outdoor transportation mode detection. This contribution shows that aggregating GPS and GIS spatial and temporal classification features with that of the mobile phones accelerometer sensor improves transportation mode detection accuracy. Specifically, it highlights that accelerometer sensor is sufficient for indoor transportation mode activity recognition.

1.5.3 Parking status detection

The availability of the vacant parking spaces can be calculated by external sensors such as those installed in the road surface. For example, in San Francisco, several parking projects have been initiated [17, 18] that utilize externally implanted sensors. However, this strategy is expensive to deploy and maintain. For example, in one project that covers 8000 parking spaces the cost was over USD $23 million [17, 19].The proposed solutions in this dissertation only require sensors on a mobile phone and not external sensors. External sensors may underperform in extreme weather. For example, in heavy snow these external road surface implanted sensors may be covered. Using mobile phones is cheaper, more convenient, and more flexible.

This dissertation contributes three methods for a PSD to detect parking status of street parking slot:

1. *Connection to in-vehicle Bluetooth*. This method utilizes the Bluetooth connection between the driver's mobile phone and the car to detect parking/deparking activities. Nowadays more and more cars have in-vehicle Bluetooth capability, which allows the owner of a car to register her mobile phone to the in-vehicle Bluetooth system. Once the mobile phone is registered, whenever the owner is inside the car and the engine is started, the mobile phone is automatically connected to the in-vehicle Bluetooth system. On the other hand, when the engine is stopped or the owner goes away from the car for a short distance (~10 meters), the Bluetooth connection between the mobile phone and the car is broken. Thus, if the Bluetooth connection between the driver's phone and the car is broken, then it can be inferred that the driver parks the car. If the Bluetooth connection is established, then it can be inferred that the driver deparks.

2. *Transportation mode monitoring*. This method employs a transportation mode detection algorithm to track the transportation mode of the driver. It monitors the transitions of transportation mode and infers parking status from a sequence of transitions. For example, if the transition sequence *driving* → *stationary* → *walking* is observed, then the PSD infers that the driver parked at the stationary point. Similarly, if the transition sequence *walking* → *stationary* → *driving* is observed, then the PSD infers that the driver deparked at the stationary point.

3. *Pay-by-phone Piggyback*. This method takes the advantage of the pay-by-phone parking payment service. This service allows a driver to pay for parking by entering to her mobile phone the parking slot number (posted on parking meters) and parking duration [20, 21]. The information is then submitted from the mobile phone to a payment service system. The pay-by-phone may help parking detection in the following way. With the mobile phone user's agreement, the PSD (which resides on the same phone as the pay-by-phone application) monitors the user's usage of pay-by-phone service. When the user submits the payment, the PSD infers that the user has parked. The PSD may possibly overhear the parking duration input and thus predict the time of deparking. The pay-by-phone service has been available in over 30 cities in US including San Francisco, Chicago, Washington DC, etc. (see [20, 21]).

1.5.4 Street parking availability estimation

Contributions on street parking availability estimation includes (1) an algorithm that can construct the historical availability profile (i.e., mean and variance of parking) for a given street block, (2) Algorithms for estimating in real-time the current parking availability on a street block, (4) Experimental evaluation of the algorithms using actual real-time street parking data from SFPark.org, and (5) Theoretical underpinnings of the proposed techniques. Apart from studying the effectiveness on real time parking estimation of using the historical statistics and the scaled real time observations solely, the dissertation discusses the combination of both the historic statistics and the real time observation. The combination strategies are: (1) Using a fitted weight scheme and (2) Using an adaptive Kalman Filter scheme. These four schemes are compared with the true real time parking availability in the evaluation of the contributions.

1.5.5 Privacy-aware mobile sensing

Further contributions of this dissertation are in the area of privacy preservation mobile systems where we developed and evaluated a suite of algorithm called *MobiPriv*. *MobiPriv* is compared against previously proposed privacy preservation algorithms for mobile location based systems [9, 22]. Results indicate that *CLK* (i.e. CloaklessK) and CK (i.e. CloakedK) substantially improved the success rate in location based privacy systems. Even with high quality of service requirement, *CLK* and CK can achieve high success rate. With previously proposed algorithms, under certain conditions, such as high quality of service requirement, or high anonymity level, these privacy systems discard a number of mobile requests that cannot be anonymized. On the other hand, *MobiPriv* provides a “guaranteed service“, and all the mobile requests sent to our anonymization server can be anonymized. There is also no temporal delay for *K-1* other clients to become available in the proposed work. *MobiPriv* algorithms have a good run time performance. In particular, for low anonymity level, *MobiPriv*’s *CK* supersedes the other privacy algorithms such as Casper [9] and PrivacyGrid [22] in run time performance. This run time result conforms to the algorithm complexity asymptotic analysis. The communication cost of *MobiPriv* algorithms is higher. However, a communication cost reduction strategy is used in *MobiPriv*’s *CK*. These results indicate that previously proposed snapshot privacy algorithms, such as Casper and PrivacyGrid cannot overcome the privacy challenges in continuous queries while *MobiPriv* can.

1.6 Applications

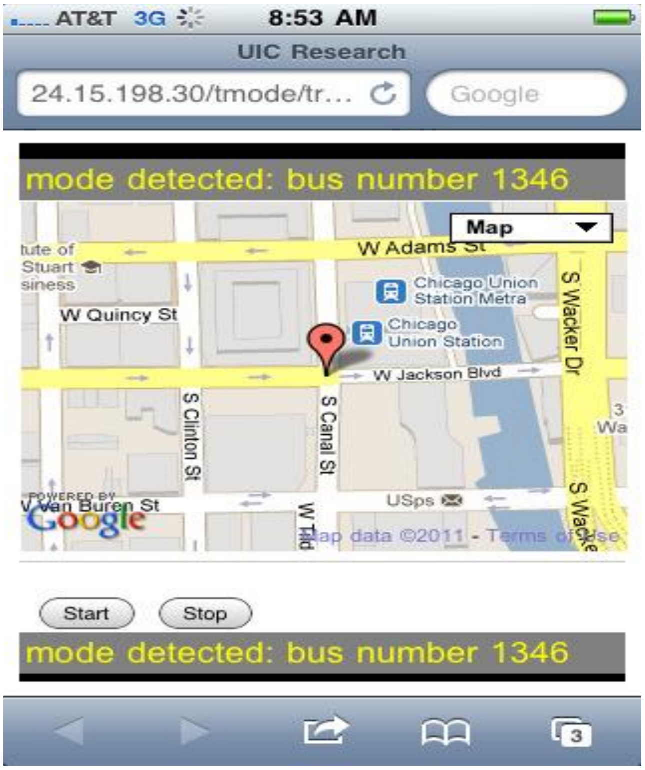
This dissertation describes possible applications of automated human activity detection from smartphones and maps (i.e. geospatial knowledge). The first application is travel demand survey dissemination and collection. Traditionally, travel demand surveys are done using telephones, interviews, and questionnaires. These traditional collection strategies rely on manual labeling of data after a trip. Thus, inaccuracies are introduced. For example, a traveler may not recall the exact time or location that she boarded a bus. Consequently, automated transportation mode detection can produce more accurate travel demand surveys since automating the detection of transportation modes provides more realistic travel demand picture [23, 24, 25].

Another application of activity recognition is calorie consumption tracking [3, 4, 26, 27]. With knowledge of both indoor and outdoor transportation activities, the amount of calories that a traveler burns can be determined. For example, if one is stationary they burn fewer calories than if they are walking or biking. Likewise a person that is utilizing an elevator may burn fewer calories than if they climbed a stairs. Thus, with knowledge of both indoor and outdoor transportation modes, an estimate of the number of calories burnt can be ascertained. Additionally, a person's carbon dioxide emissions can be estimated [2] as a function of transportation mode and distance travelled.

Another application of transportation mode detection is in the field of mobile advertising. With knowledge of the traveler's transportation mode, more accurately customized advertisements can be sent to the traveler's mobile phone. For example, if we detect that Mary is driving a car, we can send her gas coupons or vehicle service specials.

Crowdsourcing real-time traffic information from the public is done by many traffic and navigation service providers such as Google or NAVTEQ. For these companies, it is important to differentiate between transportation modes in order to determine how to utilize the crowd-sourced probe data. For example, for real time road traffic condition determination, probe data from people riding trains is not useful. On the other hand, probe data from people driving cars should be handled differently from probe data of bus travelers in order to provide accurate real rime traffic conditions. This difference in the processing of bus and car data makes sense, buses stop at bus stops regardless of traffic. While, cars continue along the road geometry in the absence of traffic.

In this dissertation, if we detect that a traveler is travelling by bus, we can further deduce which bus the traveler is currently riding (see Figure 2).



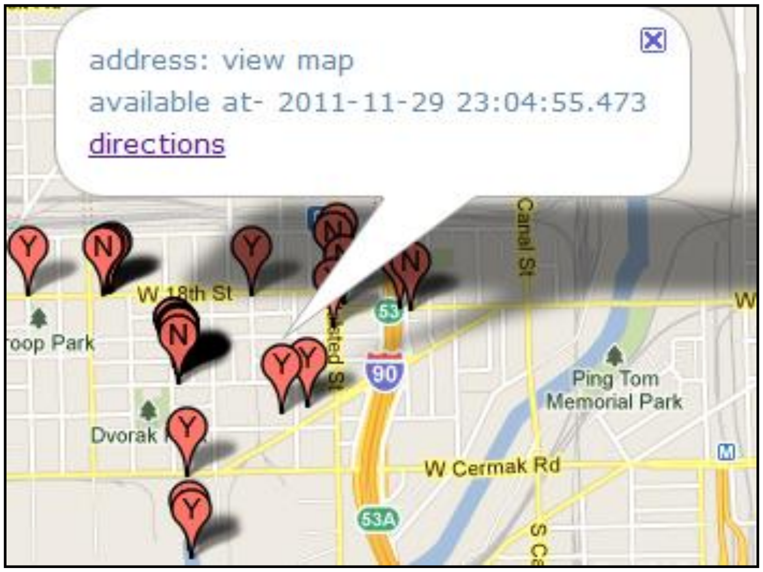
mode detected: bus number 1346

mode detected: bus number 1346

Figure 2- Fine granularity of bus mode detection

Applications of parking activity detection are important for several reasons. In one business district of Los Angeles, researchers found that vehicles searching for parking created an equivalence of 38 trips around world, produced 730 tons of carbon dioxide, and burned 47,000 gallons of gasoline [28] in one year. To deal with the parking problem, cities such as San Francisco have invested millions of dollars on smart parking infrastructures.

Using parking status detection such as Bluetooth, pay by phone piggy back, or transportation mode transitions, a map of parking availability can be created (see Figures 3). In Figures 3, Y means "yes" a parking spot is available and N means "no" a parking spot is not available.



address: view map
available at- 2011-11-29 23:04:55.473
directions

Figure 3- Map of street parking availability (parking spot level)

Observe that the location of the detected parking and deparking events can be inaccurate due to several reasons (e.g. GPS accuracy). To alleviate these problems, instead of reporting parking availability on a parking spot level, one can consider aggregating these inaccurate parking activities and then report parking availability on a street block level. Street block reporting is the role of the parking availability estimator. With parking availability estimation, applications that predict the number of available street parking spaces on a given street block are enabled using only the mobile phones as PSDs.

See Figure 4, for the application example of constructing availability at the street block level. The red polylines indicate that a candidate street block have low parking availability, light blue indicates that the parking availability is high, and dark blue indicates average parking availability.

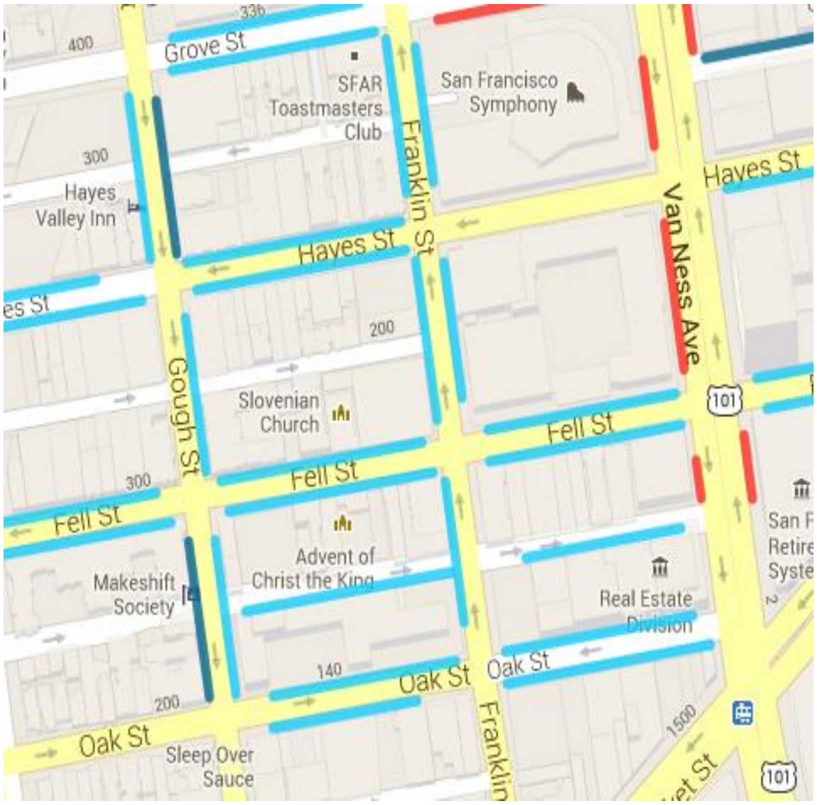


Figure 4-Map of street parking availability (street block level)

2. LITERATURE REVIEW

The techniques for human activity detection based on previous solutions are discussed in three subsections: those implemented in commercial devices, custom hardware, and mobile phones. The related work on parking status detection, parking availability estimation, and privacy aware mobile sensing is discussed separately.

2.2 Activity detection

In this section of the dissertation, we discuss activity recognition on commercial devices, custom hardware, and mobile phones.

2.2.1 Commercial devices

Commercial devices for activity recognition vary in the type of context they can detect and the kind of sensors they utilize. The pedometer is one of the most common devices for activity detection [26]. It computes the number of steps a person makes by observing upward and downward movements and consists of sensors that may include accelerometer, magnetic switch, or mechanical arm. A common pedometer implementation is found on Apple's IPod Nano [27].

Other devices such as Philip's Tracmor are sophisticated and use an accelerometer to provide a method to determine the number of calories the carrier burns. Impact Sport's ePulse [3] and Body Media's GoWear [4] also can determine the number of calories burned. However, ePulse uses a heart rate monitor and GoWear combines four sensors including accelerometer, heat flux, galvanic skin response, and skin temperature.

In general, the above commercial devices provide rough activity detection such as calories burned or distance travelled. This proposed work is different, as it detects a finer granularity of context including both indoor (stairs, escalator, elevator) and outdoor transportation modes (car, bus, train, walk, stationary).

2.2.2 Custom hardware

Different approaches to infer activities along with transportation modes using custom hardware have been considered. For example, [29] considered a single accelerometer to infer the transportation modes walking, running, and stationary. Several researchers such as [30, 31] combined multiple accelerometers placed on different sections of the human body for activity recognition. The proposed work is different, because a single accelerometer and GPS on a traveler’s mobile phone is considered.

The work of [32] utilizes a single sensing unit with multiple sensors including audio sensors, accelerometer, and barometric pressure sensors. Recent implementations also consider using Wi-Fi/cell tower fingerprints and signal strength for transportation mode detection [33, 34]. In [35, 36], over 20 sensors that are wearable on the human body are used. We believe it is highly unlikely for a person to carry that many sensors on the body in everyday life.

In [37], the authors considered GPS, accelerometer, and maps for assessing the location and intensity of travel behaviors. Our approach is different since we considered a supervised machine learning algorithm where the travelers’ mobility patterns are learnt and then recognized; while [37] does not consider machine learning. Instead, in [37] experiment participants annotate their travel behaviors in a travel log, likewise the GPS and accelerometer also record their current physical activities. Given these travel logs and sensor reports, the authors then assess the data using custom built software such as Actical and Trackstick manager for energy expenditure, activity intensity, and activity counts. GIS data such as map layers were used for displaying and understanding the traces and activities in [37], while we used GIS to create supervise learning classification features.

The work in [38] considered fuzzy rules and utilizes only the GPS sensor. Our proposed approach is different than [38], in that we aggregate multiple sensors in addition to real time and static GIS data instead of focusing solely on GPS. Additionally, [38] detects only outdoor transportation modes while this dissertation goes one step further and detects both indoor and outdoor transportation modes. Relying only on GPS makes the system limited since GPS signals can be unavailable at times. SeMiTri [39] also utilizes the GPS sensor and focuses on semantic labeling of heterogeneous GPS traces. Semantic annotation is, for example, transportation mode (walk, bike, bus, and metro) or travelers' goals such as market, work, or shopping. The proposed work considered more sensors, both indoor/outdoor transportation modes, and GIS information.

Both GPS and accelerometer units were utilized in [40] to detect skating, driving, running, and cycling. In addition to these two sensors utilized in [40], we further considered real time and static GIS data and can infer more transportation modes such as bus, car, train, walk, elevator, escalator, stairs, and stationary.

2.2.3 Mobile phones

Mobile phones are omnipresent, recent advances in memory and processing capabilities make them wearable computers. Using the mobile phone's platform for activity recognition makes the system convenient because a person has no need to carry additional hardware.

In [12, 13], the authors proposed an unsupervised learning technique to detect the traveler's transportation mode. The modes considered in [12, 13] include bus, car, and walk. In addition to identifying the transportation modes used, [12, 13] can identify the traveler's goals, such as trip purpose and trip destination. The GPS sensor along with the travel history of the traveler such as where they parked their car or board a bus is used in [12, 13]. The proposed work is different since the travel history of a person such as where they parked their car is not considered.

Additionally, we use a supervised learning model and consider more sensors (i.e. accelerometer). Also, the present work can detect both indoor and outdoor transportation activities.

The work of Zheng et al [14, 15] used only the GPS sensor to infer transportation mode. They proposed a post processing strategy and a robust set of classification features that make the model rigid to traffic and weather changes. The present work is different because we consider more sensors and transportation network information. Also, we focus on both indoor and outdoor transportation modes, while [14, 15] focuses on a subset of outdoor transportation modes.

The Jigsaw sensing system proposed in [41] balances the performance needs of the application and resource demands of continuous sensing on mobile phones. Our work is different since [41] focuses solely on outdoor transportation mode and does not consider GIS information such as real time bus locations and bus stop spatial properties. Also, [41] does not distinguish between motorized transportation modes such as buses and trains. Another energy aware mobile sensing transportation activity recognition system is [42]. The authors of [42] combined the minimal set of sensors such as GPS, accelerometer, microphone, or Wi-Fi. The modes that [42] detects are still, walking, and vehicle. I.e. [42] does not distinguish between types of vehicles nor does it detects indoor modes such as elevator or escalator.

The PEIR system [1] considered a Hidden Markov Model (HMM) for transportation mode detection and utilizes GPS, GSM, and open source weather and traffic information. In addition to transportation mode detection, the work in [1] also measures environmental impacts (e.g. carbon impact) and exposures (e.g. smog). This dissertation is different from [1], in that we considered the accelerometer sensor and a wider variety of transportation modes.

The work in [43] focuses on predicting bus arrival times from crowd-sourced information obtained from the mobile phones of bus riders. In [43], the authors utilize the accelerometer sensor on travelers' mobile phones to distinguish between buses and trains, thus determining the source of the crowd-sourced information. In our work, we used more sensors, GIS data about the transportation network, and we detect both indoor and outdoor modes. In addition, we distinguish among more categories of motorized transportation modes.

Alternatively [2] estimates real time CO_2 emissions using inertial information obtained from the mobile phone's accelerometer sensor. In order to quantify the amount of CO_2 emissions, the authors computed a function of transportation mode and distance. The transportation modes detected in [2] are outdoor modes. Our work is different, we utilize the GPS sensor, GIS, and also considered indoor transportation modes such as elevators and escalators. The state of the art that used both GPS and accelerometer sensors on the mobile phone for transportation mode detection is [11]. However, [11] does not distinguish between motorized transportation modes such as car, bus, and train. Distinguishing between motorized modes is a realistic and important problem for navigation and real time traffic providers such as Google or Nokia Location and Commerce (formerly NavTeq). These companies collect data from probes, data coming from travelers on buses, cars, and trains are handled differently for real time traffic analysis. Thus, while [11] only detects still, walk, run, bike, and general motorized modes. This work distinguishes between motorized modes such as car, bus, and train, and also considers indoor modes.

The state of the art that combines GPS and GIS data for transportation mode detection on mobile phones is [28]. Additionally, in [28], if a traveler is traveling by public bus, the system can provide further information on which particular bus she is riding. The drawback of [28] is that it does not consider the accelerometer sensor and it does not detect indoor transportation modes such as elevator and stairs.

2.3 Street parking status detection and availability estimation

A number of approaches have been considered for monitoring parking spaces. Some approaches monitor parking spaces by color histogram classification and car feature point detection from still images captured by externally mounted cameras [44, 45]. Our solution only needs sensors on a mobile phone and not external cameras.

Parking garages use in-out counters at exit and entry points to compute the number of additional vehicles they can accommodate [45]. There exist other systems that allow travelers to access parking information and make prior reservations for parking in areas such as airports and rail stations [46, 47]. PhonePark is different and focuses on street parking.

The GPS sensor has been used in the past for parking guidance. For example, GPS guided parking is used in [48]. This dissertation solves a different problem, namely automatically detecting parking and deparking activities. The available parking spaces detected by this thesis may serve as an input to the parking guidance system.

The availability of the vacant parking spaces can be calculated by external sensors such as those installed in the parking areas, which count the number of cars which enter and exit from the parking space [17]. For example, in San Francisco, several parking projects have been initiated [17, 18] that utilize externally implanted sensors. However, this strategy is expensive to deploy and maintain. For example, in one project that covers 8000 parking spaces the cost was over USD $23 million [17, 19]. The proposed work in this dissertation is not dependent on external sensors.

Ultrasonic sensors at the top of each parking space or on vehicle side-doors [19] can be used to sense the availability or unavailability of each parking space. Our solution uses only sensors available on mobile phone to infer when and where a traveler had parked their car. Sensors implanted under road surfaces or attached to the car side-doors are expensive to deploy and maintain (e.g., [17] cost USD $500 per system for each parking space, and [19] cost USD $400 per system for each car). These sensors may underperform in extreme weather. For example, in heavy snow these sensors may be covered. Using mobile phones is cheaper, more convenient, and more flexible.

The works [49, 50, 51] are based on parking but the focus is on how to choose parking slots ideally in competitive parking settings. Thus, in these other works routing to the most optimal parking slot is the focus. This thesis is different and focuses on estimating the current street parking availability on a street block in real time.

The work in [52, 53] analyze the capacity of an opportunistic system to assist the search for parking space. Vehicles searching for parking space are equipped with sensors that allow them to sense the location and status of parking spots as they drive across the city. This information is subsequently shared upon encounters with other vehicles or submitted to a central server. The proposed work in this thesis is different and focuses on detecting and estimating the availability of street parking spaces. On the other hand, [52, 53] focus on optimizing the effectiveness of the parking search process through user oriented performance metrics, such as the parking search time, route length, and the proximity of the found or assigned parking spot to the user travel destination.

Other areas of parking research includes pursuing parking spaces and responding to pricing policies about public and private parking facilities that is studied in [54, 55]. The drivers are modeled as strategic agents who make rational decisions while minimizing the cost of the acquiring parking spots, under deterministic or probabilistic information for the overall parking demand.

Google’s approach to parking status detection is called OpenSpot [56, 57] and relies on drivers to manually report an empty parking spot when they depark or see one available. This system is cumbersome and drivers don’t feel comfortable to use their mobile phones for manual reporting while driving. The proposed approach in this thesis automates parking status detection and manual entry of available street parking spaces is not required.

2.4 Privacy aware mobile sensing

The concept of K-anonymity was originally used within the realm of relational databases [58]. The idea of location K-anonymity was first proposed by Gruteser and Grunwald in [59]. The discussion on the related work in privacy preservation mobile sensing is presented in two sections. First, location based privacy in snapshot systems is discussed. Then, location based privacy in continuous querying system is discussed. In location based snapshot systems, a location based query is submitted once to the location based server. For example, “*Where is my nearest bus stop?”* In location-based continuous queries, the same query is submitted at discrete time points. For example, “*Continuously send me Groupon coupons that are redeemable at least 2 miles from my current location?”*

2.4.1 Snapshot queries

The main shortcoming of privacy aware snapshot systems is that if K-1 other requests cannot be found, the query is discarded because the desired privacy cannot be provided. Current anonymization techniques [9, 10] are ineffective if K-1 other users or requests cannot be found. In these works, if K-1 other queries cannot be located within the QoS constraints, the query is dropped. This is where our research on snapshot system privacy is focused: on improving the success rate. Thus, in this dissertation, all requests sent to the AS can be anonymised safely without being dropped. We evaluate this measure of drop queries as success rate.

In [10], the value of K in K-anonymity is static and uniform for all mobile users in the system. The framework of a personalized value of K was first introduced in [60] by Gedik and Lui. In this model, each user defines their personalized value of K as opposed to the static approach in [10] where all the users share the same K value.

The authors in [61] neglect to use anonymization servers, instead relied on a client-server technique called “*SpaceTwist*”. In [61], the authors defined the concept of demand space and supply space. The demand space is the space yet to be explored and the supply space is the space already explored. The true location of the client is only known by the client. The client then sends a request using a false location to the server. On receiving the request, the server then replies with a response. The client incrementally sends more requests to the server increasing its supply space and reducing its demand space. When the supply space totally covers the demand space, the algorithm completes and the client is guaranteed to have received a correct response. We identify two obvious pitfalls with such an algorithm. The first is excess communication cost. Secondly, if an adversary has background knowledge that a person is at a particular spatial point re-identification is possible.

In [61, 62, 63, 64], strict client-server approaches are considered instead of using anonymization servers. Chow’s work in [10] is based on the peer to peer approach for privacy preservation. The proposed work is different and we concentrated on the trusted third party architecture with the anonymization server. Below, the four most relevant works to the proposed algorithms are discussed. The four most relevant works are *Interval Cloaking* [59], *Clique-Cloaking* [60], *Casper* (Pyramid Based) [9], and *PrivacyGrid* [22].

- *Interval Cloaking* [59] - This cloaking technique iteratively divides the region into quadrants. The region before division is called q_{prev}. If after division the quadrant that the user resides contains more than *K* users, then iterative division continues. If after division the quadrant that user resides contains less than K users, then q_{prev} is returned as the cloaking box.

- *Clique-Cloaking* [60] - A clique in a constraint graph is identified. The nodes in the graph are messages and the edges are formed between two messages if the cloaking region of the messages overlaps to include the messages. Messages can be cloaked together if they are neighbors in the graph and the K requirement can be satisfied. If messages cannot be cloaked they are expired and dropped. Also, users tend to be close to edges of the minimum bounding rectangle.
- *Casper* [60] – Casper maintains an anonymizer and a privacy aware query processor. For anonymization and cloaking, a pyramid structure is maintained. The cells in the region contain the number of mobile users active in the cell. If the current region/cell of user cannot satisfy the value of K or A_{min} then a neighboring cells are considered. Casper has a superb run time performance. However, Casper does not return the smallest cloaking region and also it is expensive (*updates and cloaking*) to maintain the pyramid structure. The same authors of [60] also introduced *TinyCasper* [65]. TinyCasper is solely for wireless sensor networks. We evaluated the proposed work against Casper [60].
- *PrivacyGrid* [22] - *PrivacyGrid* guarantees a small CR by using a *Bottom-up*, *Top-down,* or *Hybrid* approach. In the *Bottom-up* approach, the mobile user’s cell is expanded to meet the K-anonymity and L-diversity requirement. For a given cell it may expand to its immediate neighbor east, west, north or south. The next cell to be chosen is the cell with the highest mobile user count. This cell will be included in cloaking box.

The *Top-down* approach selects the largest possible cloaking box that satisfies the users QoS requirement. If K users cannot be found in this cloaking box, the query cannot be satisfied. If K users are found, then the algorithm attempts to prune the cloaking box to a smaller cloaking box and verifies if K mobile users are still present. The *Hybrid* approach makes a decision to use bottom up or top down, depending on the value of K and the QoS (i.e. spatial tolerance). The *Hybrid* scheme combines the strengths of both the top down and the bottom up approaches.

We evaluated the effectiveness of the proposed algorithms against, Casper, PrivacyGrid's Bottom up, PrivacyGrid's Top down, and PrivacyGrid's Hybrid schemes.

2.4.2 Continuous query

A continuous query is a snapshot query submitted at discrete time points. Most of the current works on privacy aware mobile location based systems are focused on snapshot queries. Since the cloaking set for the same mobile user may be different at distinct time stamps, a snapshot solution may not be sufficient in a continuous querying environment. An aggregation or intersection of multiple snapshots in a continuous query can lead to privacy breaches [66].

Work on location based continuous queries includes [7, 66]. In [6], the same set of mobile users are aggregated and submitted to the location based system by the anonymization server for each request from a mobile client. The main shortfall of this model is the possible reduction of quality of service for subsequent mobile requests from the same mobile client. The quality of service problem of [66] was addressed in [7], and [7] assumes that the transportation mode can be determined from a set of GPS points [8]. Thus, in [7], they anonymize each local and global candidate set by considering similar transportation modes using a dynamic cloaking strategy.

Since current snapshot solutions are not applicable for privacy reservation in continuous querying systems, a robust and effective model for location based privacy preservation for continuous queries is proposed in this work.

Dummies were first introduced by Kido [62]. The proposed work is different, realistic and diverse dummy request are generated on the anonymization server with respect to K-anonymity. The dummy related work in [62] uses the strict client server architecture without the anonymization server. Additionally, [62] did not consider K-anonymity or diversity constraints when generating dummies.

3. METHODOLOGY

In this chapter, we introduce the general model considered throughout the dissertation.

3.2 Summary of general model

The system consists of a finite set of *mobile nodes* t_1, t_2,t_3 … t_n and a *central servers S*. The server *S* does not facilitate communication between mobile nodes. A mobile node is a human traveler t_i with a sensing device M_i. In this dissertation, we assume that *M* is a smartphone that consists of several sensors s_1, s_2, s_3 … s_m such as GPS and accelerometer sensors. This assumption is realistic since these sensors are present on smartphones. At discrete time points, the traveler t_i submits sensor reports s_1^{rep}, s_2^{rep}, s_3^{rep} … s_m^{rep} which consists of data from the sensing devices on M_i to *S*. The data is formed as a fix set of attributes and corresponding values. An example of a report is a GPS report, whose attributes can be latitude, longitude, and timestamp.

Given these sensor reports by t_i from M_i, *S* then aggregates these reports with knowledge of the underlying transportation network. More specifically, the GPS reports are aggregated with transportation network data. Let us represent a transportation network as a set $T = \{B, R, BS\}$, where *B* is set of real time bus locations, *R* represents rail line locations, and *BS* represents the spatial properties of bus stops. Given the GPS reports and *T*, spatial and temporal mobility classification patterns are created, these patterns results from combining *B*, *R*, and *BS* with t_i*'s* location. Additionally, mobility patterns are also created from the other sensors, however data from the other sensors is not fused with the data from *T*. An instance of sensor report from t_i is cached in *S's* database. When *S* receives a time window t^{win} amount of data from t_i, the activity of t_i can then be determine probabilistically by *S* in real time.

For transportation mode detection, *S* predicts the transportation mode probabilistically instantaneously. For parking status detection by transportation mode transitions, *S* forwards the detected transportation mode to a finite state machine *F* for further processing.

Parking availability estimation takes multiple sparse and erroneous parking status detection observations and or historic parking information to make the final estimation. Historic parking information is constructed from these sparse and erroneous parking status detectors. The dissertation shows how to restrict the error when constructing historical parking information for street block.

Parking availability estimation takes into the fact the penetration ratio of the parking status detectors may be low since not all drivers will have a parking status detector. Parking availability estimation also takes into consideration that a parking status detector may have false positives and false negative parking and deparking activities. A false positive parking activity occurs when the parking status detector reports a parking or deparking activity that did not take place. A false negative report occurs when the parking status detector fail to detect the parking or deparking event.

3.3 Smartphone sensors

There are several sensors on mobile phone. Sensors include GPS, accelerometer, microphone, camera/s, gyroscope, etc. Data streams from two sensors on the traveler's mobile phone are used to detect the transportation mode that a traveler is using. The two sensors are: (1) Accelerometer, and (2) GPS. These two sensors are commonly available and accessible on mobile phones. For parking detection, we utilize two approaches, one uses the detected transportation modes, and the other utilizes the Bluetooth sensor on the mobile phone. Thus, we will discuss GPS, accelerometer, and Bluetooth.

3.3.1 Accelerometer sensor

The accelerometer measures acceleration in all three axes (i.e. x, y, and z). From a mobile phone, one can access the acceleration with gravity and also without gravity and can then compute statistical features that are unique for a particular activity. The accelerometer is useful in distinguishing between motorized and non-motorized transportation modes. However, to distinguish between different motorized modes such as car and bus using only accelerometer is challenging since the force vector values for motorized transportation modes are similar.

3.3.2 GPS sensor

The GPS sensor can provide location positioning data. From GPS, properties such as latitude, longitude, speed, acceleration, and heading change can be computed. Given these attributes, it is possible to infer a person’s context. For indoor context detection, GPS may not be suitable since accuracy decreases indoors. One pitfall of using GPS for context detection is its high power consumption. Consequently, the sampling period is important for context detection when using GPS sensors.

3.3.3 Bluetooth Sensor

Bluetooth is a wireless communication protocol used for connecting nearby devices. The neighborhood of a Bluetooth device provides useful indication of the social situation, hence motivates social community context detection. This sensor (i.e. Bluetooth) has not been applied to parking space availability detection without external beacons. Relying on Bluetooth signals on by mobile phones has challenges apart from limited range. For example, it is challenging to distinguish whether one is moving or if the environment around them is changing. Alternatively, one could implant Bluetooth sensors or Bluetooth beacons around all the available parking spaces and then communicate with the client's mobile device to detect empty or occupied parking spaces. However, we believe that this model is costly and time consuming to implement and deploy. Our propose approach of utilizing the mobile phone's Bluetooth and the in-car Bluetooth component for street parking detection is promising and less expensive to deploy than beacons. More and more cars will have Bluetooth capability, thus parking detection using Bluetooth sensors is not a shortcoming.

3.3.4 Smartphone sensor reports

A sensor report is composed of a hashed identification and the data from two mobile phone sensors: (1) GPS (2) accelerometer. Sensor reports are submitted from a mobile phone via the Internet. Bluetooth sensor reports are not submitted. Instead, the Bluetooth is used to determine parking activities locally on the device and only the spatial and temporal properties of the detected parking activity are sent to the central server.

Definition 1. **GPS sensor report.** *A GPS sensor report p_i represents data submitted from the GPS sensor embedded on a traveler's mobile device. The format of the report is <lat, lon, t, v, h, acc> where lat represents the latitude; lon represents longitude; t represents the timestamp of the sensor report; v represents the current ground speed of the device; h represents the direction of movement; and acc represents the accuracy level of the latitude and longitude coordinates.*

The measurement units of the GPS sensor report attributes are as follows: latitude (lat) and longitude (lon) are in decimal degree; current ground speed (v) is in meters per second; direction of movement (h) is in degrees counting clockwise from true north; accuracy level (acc) is in meters; and time t is in seconds.

Definition 2. GPS trace. *A GPS trace T is a sequence of GPS sensor reports,* $T = p_0 \rightarrow p_1 \rightarrow \cdots \rightarrow p_k$*, where the timestamps in the sequence strictly increase.*

Definition 3 - Accelerometer sensor report. *An accelerometer sensor report ai represents data submitted from the accelerometer sensor embedded on a traveler's mobile phone. The format of the report is <x(i), y(i), z(i)> where x(i), y(i), and z(i) represents the acceleration in m/s^2) in the x, y, and z axes, respectively.*

3.4 Geospatial data (transportation network)

Our algorithms fuse data from the sensor reports with data from the transportation network to create the classification feature vector. Specifically, we evaluated the proposed algorithms in the city of Chicago, Illinois, USA, we consider: (1) real time location of public passenger buses, (2) rail line spatial information, and (3) public passenger bus stop spatial data. Below we discuss the transportation network that was utilized in this work.

Definition 4 - Road Segment *– A road segment r is a directed edge in the road network graph with two end points r.start and r.end and traffic flow is from r.start towards r.end.*

Definition 5 - Road Network *– A road network G is a directed graph G = (V, E), where V is a set of nodes representing the end points of road segments and E is the set of edges (i.e. r).*

Definition 6 - Path *– A path p is a sequence of connected road segments* r_1 , r_2, r_3,.., r_n *and* $r_{(g+1)}.start = r_g.end$, $1<=g<n$

3.4.1 Real time bus location

The real time locations of public passenger buses are used in the proposed activity recognition detection system when available. Real time public transit tracking is performed in many cities such as London, New York, San Francisco, Toronto, Washington, and Chicago. In the proposed work, the traveler’s location is fused with real time bus locations to create mobility patterns that will be used for training the machine learning model. Let the location of the mobile user at time *t* be represented as p_t^{loc}, based on the GPS sensor report. Also, let the *m* buses in the city be bus_1 to bus_m, where $bus_{x.t}^{loc}$ is the location of bus bus_x at time t. Below, line 1 shows the mobile user’s location trace. Line 2 to line 5 represent the location traces of all the *m* buses (bus_1, bus_2, bus_3…bus_m).

From this schema, mobility patterns of the traveler and public passenger buses at time *t* can indicate the mode of transportation.

3.4.2 Bus Stops

Spatial information about public passenger bus stops is available in several cities. Our propose activity detection algorithms will use these bus stop spatial properties in conjunction with the traveler’s GPS reports to create mobility patterns that can identify in real time the mode of transportation being used. For example, if one is traveling by bus, intuitively, they should have low speeds close to bus stop locations.

***Definition 7* – Bus stop** – *A bus stop in this work is represented as a location point b [x, y] where x is the latitude and y is the longitude.*

3.4.3 Rail Lines

In the proposed algorithms, spatial rail line information is aggregated with GPS sensor data to create classification features. For example, from the traveler’s GPS location and the location of rail lines, mobility classification feature such as the Euclidian distance between the traveler's mobile device and closest rail line can be computed.

***Definition 8 – Rail line** – A rail line is a polyline with continuous location points* $[x_1, y_1]$, $[x_2, y_2]$, $[x_3, y_3]$ ..... $[x_n, y_n]$ *where* $[x_i, y_i]$ *is the location of the* i^{th} *point and x, y is the latitude and longitude respectively.*

3.5 Supervised machine learning model for activity detection

To identify the transportation mode of a traveler, we follow the general principle of supervised learning from mobile sensor traces. First, labeled sensor (i.e. GPS and accelerometer) data is collected from the traveler's mobile phone. The initial data labeling is done by the travelers from their mobile phones where they annotate the transportation mode that corresponds to their trip and sensor reports. This labeled sensor data is then fused with GIS information about the transportation network to create classification features (see Figure 5). For example, using the GPS sensor reports, the Euclidean distance to the closest bus or rail line can be computed. For another example, from the accelerometer sensor, the correlations in accelerations in the x, y, and z planes are used as features. With these classification features, a machine learning model is trained with the labeled sensor data and GIS information combined. Then, when the transportation mode is to be determined, unlabeled sensor data (i.e. GPS and accelerometer data) from the traveler's mobile phone is fused with GIS information to create the same features that were used for training. These features are then fed to the already trained classification model. This trained classification model can then determine the current activity of the traveler in a probabilistic format in real time.

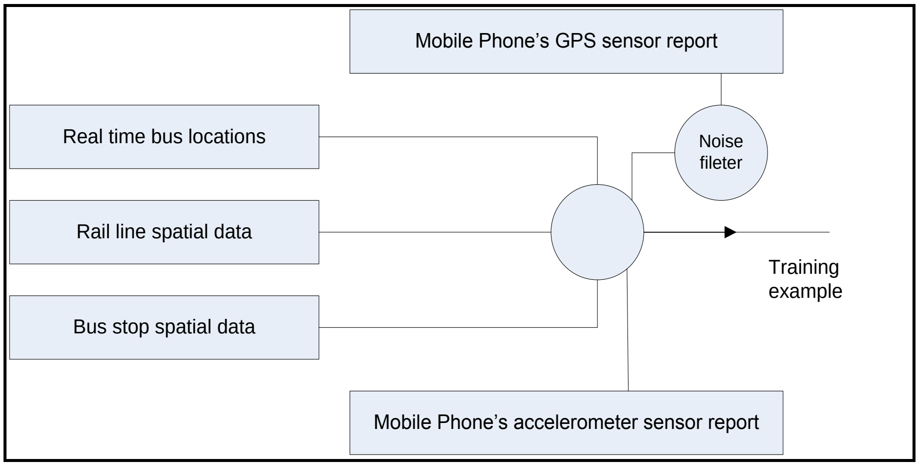


Figure 5 – Classification examples

3.6 Street parking

In this dissertation, a parking system developed is called PhonePark [28]. PhonePark software system considers the mobile web as user interface and consists of two components, namely *parking status detectors* (PSDs) and the *parking availability estimator* (PAE) (see Figure 6). A PSD runs on a mobile device. It detects when and where a driver parks/deparks and whether she pays for parking. When the PSD recognizes a parking or deparking activity, it submits a report to the PAE component indicating that a parking space is occupied or released[1]. The PAE component aggregates the reports of individual PSDs to estimate the number of available parking slots in each street block. For estimation, PAE considers the facts that not all drivers are equipped with the PSD component (i.e., the market penetration ratio of PhonePark is not 100%) and the mobile device can have false positive and false negative possibilities for parking detection. Furthermore, to compensate for the inaccuracy of PSD observations, it combines PSD observations with historical parking statistics to get the final estimate. Historical parking statistics is obtained from

---

[1] The submission of a parking/deparking activity can be anonymized and incentivized. In general, privacy and incentives are orthogonal issues that are not addressed in this paper.

the proposed HAP construction algorithm while PSD observations are obtained from the drivers with the PSD devices.

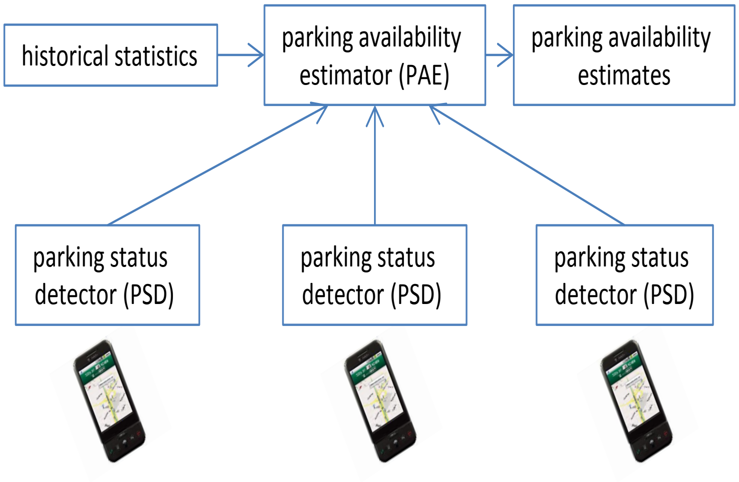

historical statistics
parking availability estimator (PAE)
parking availability estimates
parking status detector (PSD)
parking status detector (PSD)
parking status detector (PSD)

Figure 6-Components of the PhonePark system

An observation of the PSD may be incorrect due to many reasons. For example, the transition sequence *car* → *stationary* → *walking* may be the result of a passenger being dropped off (as opposed to a driver parking). Likewise, Bluetooth pairing or pay-by-phone piggyback can have errors. These errors are simulated probabilistically as false positives and false negatives.

All the errors of PSD observations propagate to the PAE component when PAE aggregates these observations to estimate the number of available parking spaces.

4. OUTDOOR ACTIVITY DETECTION

In ubiquitous and context aware computing, understanding the mobility of a client from sensor data is an important area of research. The transportation mode, such as walking, cycling, or train denotes some characteristics of the mobile user's context. With knowledge of a traveler's transportation mode, targeted and customized advertisements may be sent to the traveler's device. For example, if we discover that Alice is driving by car, the system may send her gas coupons or vehicle service specials.

Another motivation for transportation mode detection is transportation surveys. Travel demand surveys have taken multiple formats, such as telephone interviews and questionnaires. These data collection strategies rely on manual labeling of data after the trip, and thus, inaccuracies are introduced. For example, a traveler may not recall the exact time that she/he boarded a transportation mode. Using GPS devices is more reliable for reporting accurate location, trip time, and trip duration [23, 24, 25]. Hence, if the precise transportation modes of individual users are recognized, it is possible to provide a more realistic travel demand picture.

Many GPS trace sharing social networks has been implemented [67, 68, 69, 70]. These social networks enable friends to upload and share their GPS traces. Knowledge of transportation mode, added to these GPS traces, will enable the users to reflect on their past motion more meaningfully. It also allows users to obtain additional information from their friends' travel experience. Additionally, awareness of transportation mode of a user may help to determine the user's carbon footprint, or track the amount of calories burnt.

Another application of transportation mode detection is crowd-sourced real-time traffic information in which traffic speeds are aggregated from probes such as mobile phones carried by travelers. Transportation mode detection enables the aggregation system to filter out the speed data submitted by non-motorized travelers or travelers on trains.

For outdoor transportation activity recognition, the GPS sensor on the traveler's mobile phone is used in conjunction with GIS data on the underlying transportation network.

4.2 The GPS and GIS model for outdoor activity detection

This proposed model is based on the GPS sensor on the smartphone and geospatial knowledge (i.e. GIS or transportation network) for outdoor transportation mode detection [8]. The transportation modes considered are bus, car, above-ground train, bike, walk, and stationary. In general, our proposed algorithm is a supervised learning mechanism with two stages. In stage 1 (learning stage), the data from the GPS sensor report is merged with the transportation network data and labeled ground truth. This data is used to create a classification feature set that we use to train our classification model. In this stage, mobile devices submit GPS sensor reports every *t* seconds, where *t* is a system parameter. These incoming sensor reports are labeled with the corresponding transportation modes. Then, in stage 2 (inference stage), to determine a traveler's transportation mode, we first extract the same classification features as in stage 1. Subsequently, given the features, the classification system predicts the transportation mode of the traveler in a probabilistic format. Specifically, our mode detection algorithm fuses inputs from the mobile devices' GPS receivers with real time locations of buses, rail line and bus stop location data. GPS technology is a built-in feature of many mobile devices, such as IPhones, BlackBerrys, and Android phones. Given a GPS trace of a traveler, one way to build the classification model is as follows. For each GPS sensor report in the trace, various features including the closest Euclidian distance to rail lines, closest Euclidian distance to buses and bus stop closeness rate are computed. Mean speed, heading, and acceleration are also obtained over a time window. These features form a sensor feature vector. The feature vector, plus the transportation mode label of the associated time interval, forms a training example. In this way, a training set is constructed. This procedure is illustrated by Figure 7.

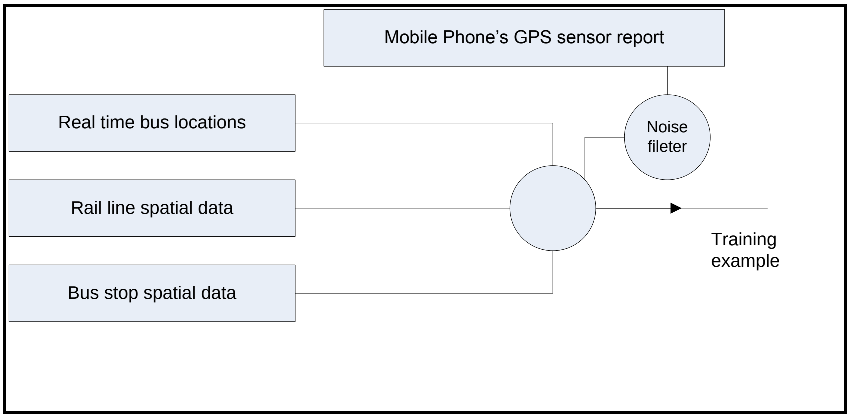


Figure 7 - GPS/GIS activity detection model

4.3 Classification features for outdoor transportation mode sensing

This section deliberates the classification features used in the proposed outdoor transportation mode sensing classification system. Additionally, the motivation for each feature, and the algorithm used for calculating the feature values are discussed. In this thesis, the classification features related to motorized transportations: (1) average bus location closeness, (2) candidate bus location closeness, (3) average rail line trajectory closeness, and (4) bus stop closeness rate. This is in addition to other features such: average accuracy of GPS coordinates, average speed, average heading change, and average acceleration. In the rest of this section we describe each of the novel features and the traditional features.

4.3.1 Average accuracy of GPS coordinates

The estimated horizontal accuracy is a measure of the confidence on the location reported by the GPS sensor; it is a component of a GPS sensor report (see Definition 1). The accuracy is reported in meters. Different transportation modes should have different estimated accuracies. For example, traveling by aboveground trains should have worse accuracy than walking, since walking has a clearer view of the GPS satellites in the sky. Additionally, we consider the average accuracy for a set of GPS reports as a feature, instead of the instantaneous accuracy. Taking the average accuracy is more realistic since the GPS system may introduce uncertainties. Let $\{p_1, p_2, p_3, p_4 \ldots p_n\}$ be a finite set of GPS sensor reports submitted from the traveler's mobile device within a time window.

$$Average\ accuracy = (\textstyle\sum_{i=1\ to\ n} p_i^{acc}) / n \qquad (1)$$

where p_i^{acc} is the estimated accuracy of the reported GPS position.

4.3.2 Average speed

In terms of speed, we use the speed value returned by the GPS sensor when it is available; this is more accurate than calculating the speed from consecutive GPS location points [11]. Otherwise, if the direct speed is not available, it can be computed from consecutive location changes. For a sequence of GPS reports we compute the average speed. This feature has been used in many existing works.

Let $\{p_1, p_2, p_3, p_4 \ldots p_n\}$ be a finite set of GPS sensor reports submitted within a time window.

$$Average\ speed = (\textstyle\sum_{i=1\ to\ n} p_i^{v}) / n \qquad (2)$$

where p_i^v is the current ground speed obtained from the GPS sensor report.

4.3.3 Average heading change

The heading is the direction from true north. For a set of GPS reports, we compute the average heading change. The heading change is an important feature for distinguishing between motorized and non-motorized transportation mode. This proposed classification feature is different from the heading feature in [14] because we compute the average heading change whereas [14] computes the heading change rate. The heading change rate in [14] is defined to be the number of times the heading change exceeds a certain threshold. It is computed as the ratio $|P_c|$ / distance, where $|P_c|$ represents the number of points where the traveler changes heading beyond the heading threshold. The heading change rate as defined in [14] cannot be used to distinguish between transportation modes with heading change rate below the chosen heading threshold. Let $\{p_1, p_2, p_3, p_4 \ldots p_n\}$ represent a finite set of GPS reports submitted within a time window.

$$\textit{Average heading change} = \left(\sum_{i=1 \text{ to } n} |p_i^h - p_{i-1}^h| \right) / n \qquad (3)$$

$$\forall 2 \le i \le n$$

where p_i^h is the direction from true north included in the GPS sensor report.

4.3.4 Average acceleration

Let $\{p_1, p_2, p_3, p_4 \ldots p_n\}$ be the finite set of GPS reports submitted within a time window.

$$p_i^{acceleration} = (p^v_i - p^v_{i-1})/(p^t_i - p^t_{i-1}), \ \forall 2 \le i \le n$$

$$\textit{Average acceleration} = \left(\sum_{i=1 \text{ to } n} p_i^{acceleration}\right) / n \qquad (4)$$

where $p_i^{acceleration}$ is the acceleration of the mobile device.

4.3.5Bus location closeness

This feature aggregates the traveler's GPS location with the real time locations of public passenger buses. Bus location closeness is useful for determining if the mobile device is on a bus or not. We develop two algorithms to determine if a mobile user is traveling by bus; (1) *average bus closeness*, and (2) *candidate bus closeness*. Let the location of the mobile user at time *t* be represented as p_t^{loc}, based on the GPS sensor report. Also, let the *m* buses in the city be bus_1 to bus_m, where $bus_{x.t}^{loc}$ is the location of bus bus_x at time t. Below, line 1 shows the mobile user's location trace. Line 2 to line 5 represent the location traces of all the *m* buses (bus_1, bus_2, $bus_3 \ldots bus_m$).

1 - p_1^{loc}, p_2^{loc}, p_3^{loc}, $p_4^{loc} \ldots p_n^{loc}$
2 - $bus_{1.1}^{loc}$, $bus_{1.2}^{loc}$, $bus_{1.3}^{loc}$, $bus_{1.4}^{loc} \ldots bus_{1.n}^{loc}$
3 - $bus_{2.1}^{loc}$, $bus_{2.2}^{loc}$, $bus_{2.3}^{loc}$, $bus_{2.4}^{loc} \ldots bus_{2.n}^{loc}$
4 - $bus_{3.1}^{loc}$, $bus_{3.2}^{loc}$, $bus_{3.3}^{loc}$, $bus_{3.4}^{loc} \ldots bus_{3.n}^{loc}$
…
5 - $bus_{m.1}^{loc}$, $bus_{m.2}^{loc}$, $bus_{m.3}^{loc}$, $bus_{m.4}^{loc} \ldots bus_{m.n}^{loc}$

4.3.6 Average bus closeness (ABC)

From GPS sensor reports $\{p_1, p_2, p_3, p_4 \ldots p_n\}$ that are submitted within a time window,

we obtain the set of locations points $\{p_1^{loc}, p_2^{loc}, p_3^{loc}, p_4^{loc} \ldots p_n^{loc}\}$. For each location point p_t^{loc}, we compute d_t^{bus} as the Euclidian distance between p_t^{loc} and the closest bus $bus_{x.t}^{loc}$ at time *t*. Subsequently, given d_t^{bus}, we calculate the feature *average bus closeness* (ABC), as the average Euclidian distance of (d_1^{bus}, d_2^{bus}, d_3^{bus}, $d_4^{bus} \ldots d_n^{bus}$), for the set of GPS sensor reports $\{p_1, p_2, p_3, p_4 \ldots p_n\}$.

$$ABC = \left(\sum_{i=1 \text{ to } n} d_i^{bus} \right) / n \qquad (5)$$

This feature is used to capture whether the traveler is traveling via bus transportation mode.

4.3.7 Candidate bus closeness (CBC)

First, we obtain the set of locations points $\{p_1^{loc}, \ p_2^{loc}, \ p_3^{loc}, \ p_4^{loc} \dots p_n^{loc}\}$ from GPS sensor reports $\{p_1, p_2, p_3, p_4 \dots p_n\}$ that are submitted within a time window. For each location point p_t^{loc}, we compute the Euclidian distance $d_{j.t}^{bus}$ $1 \leq j \leq m$ to each bus bus_j, in the set of all buses $\{bus_1, bus_2, bus_3 \dots bus_m\}$ at time t. Then, for each bus bus_j, we compute the total Euclidian distance D_j over the time window as follows.

$$D_j = \sum_{t=1 \text{ to } n} d_{j.t}^{bus} \qquad 1 \leq j \leq m \qquad (6)$$

Given D_j for all the m buses, we compute CBC as follow.

$$CBC = min\,(D_j) \qquad 1 \leq j \leq m \qquad (7)$$

The classification feature CBC is the minimum D_j value. Using the CBC feature requires more memory than the ABC counterpart, since the Euclidian distance from the device to every bus in the city needs to be computed and stored for each GPS sensor report. For ABC we only compute the distance to the closest bus.

4.3.8 Rail line trajectory closeness

This classification feature relates the traveler's GPS location with spatial data representing the rail network. This feature may be useful to detect if a person is travelling on an aboveground train. For underground trains (subways), since GPS does not work well underground, this feature may not be effective. We do not consider subways in this work. The Euclidian distance d_i^{rail} between a person's mobile device and the closest rail line is computed for each GPS sensor report p_i. We then calculate the classification feature average rail location closeness (ARLC) as follows. Let $\{p_1, p_2, p_3, p_4 \dots p_n\}$ be a finite the set of GPS reports submitted within a time window.

$$ARLC = \sum_{i=1 \text{ to } n} d_i^{rail} \ / \ |n| \qquad (8)$$

4.3.9 Bus stop closeness rate

This classification feature relates the traveler's GPS location with spatial bus stop data. First, from experiments we determine a *bus stop closeness threshold*. This threshold is a Euclidian distance measure and may be used to concur if a person is at a bus stop. We calculate the classification feature BSCR (*bus stop closeness rate*) as follows. Let $\{p_1, p_2, p_3, p_4 \ldots p_n\}$ be a finite set of GPS sensor reports submitted within a time window. BSCR stands for the number of GPS sensor reports p_i, whose Euclidian distance $d_i^{busstop}$ to the closest bus stop, is less than *the bus stop closeness threshold* within a unit time.

$$BSCR = |\,PS\,| \,/\, window\ size \qquad (9)$$

where $PS = \{p_i \mid p_i\ \varepsilon\ \{p_1, p_2, p_3, p_4 \ldots p_n\},\ d_i^{busstop} < bus\ stop\ closeness\ threshold\}$.

<u>Experiments to determine bus stop closeness threshold value</u>

Below, we explain how to obtain the *bus stop closeness threshold* value. For experiments, a traveler carried a mobile device and boarded a CTA bus. We then measured the Euclidian distance to the closest bus stop from the traveler's device. From over 450 GPS sensor reports, we plot the graph of Figure 8. The vertical axis represents the Euclidian distance to the closest bus stop. The horizontal axis represents the corresponding GPS sensor report number. From Figure 8, we observe how the mobile device's Euclidian distance to the closest bus stop fluctuates during the travel. When the traveler on a bus is at the bus stop, the distance is at a minimum. As the bus moves away from the bus stop, the distance increases. It peaks at the midpoint between two bus stops. Afterward, it decreases and reaches a new minimum when the bus reaches the next bus stop.

When the traveler is traveling via bus mode and at a bus stop, the Euclidian distance to the closest bus stop is less than 13 meters. Thus, for *bus stop closeness threshold*, we used a value of 13m.

For BSCR, we compute the number of times the Euclidian distance to the closest goes below the *bus stop closeness threshold* per unit time. We believe that if a traveler is traveling by bus, the BSCR should be greater than if they are not travelling by bus. We also evaluate the effectiveness of BSCR on predicting the outdoor transportation mode in the proposed work.

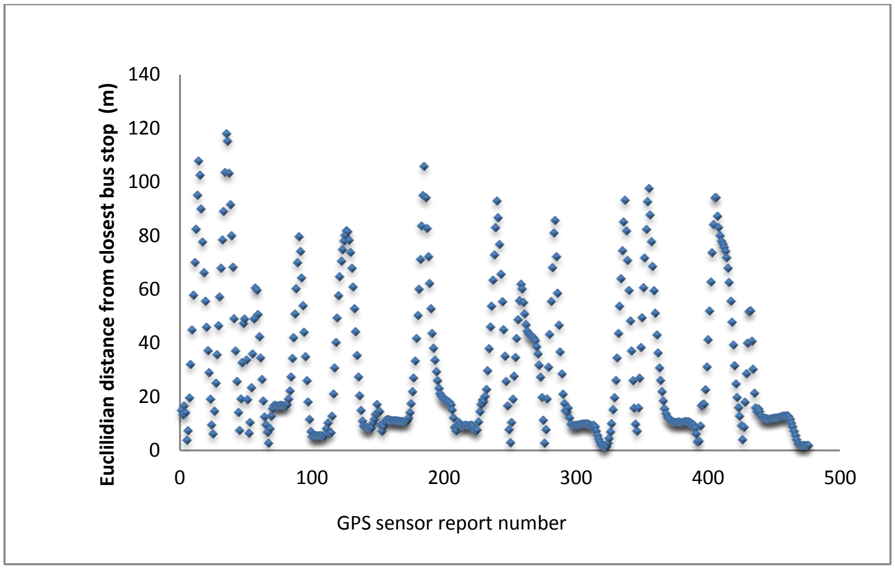


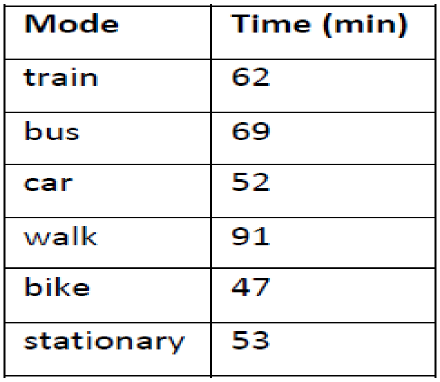
Figure 8-Euclidian distance to closest bus stop while one rides a bus

4.4 The application environment

The environment consists of a set of travelers with smartphones with GPS sensors in the city of Chicago. More specifically, each traveler has an Apple IPhone 3G, HP IPAQ, or Samsung Galaxy. GPS reports are submitted from the device every 15 seconds and a window size of two 30 seconds is chosen as the period of classification. The data collection experiment period was for three weeks.

The transportation network considered for this application environment is that of Chicago Transit Authority (CTA). CTA has over 1,700 buses in service that operates over 140 routes. On an average weekday, 1.7 million rides are taken [71]. The real time locations of the buses are updated every 20-30 seconds, and the data is available freely to the public as an API in XML format. Information available about the CTA buses includes: route, latitude, longitude, final stop, bus identification, and direction of over 1000 buses. Additionally, the CTA services over 11,577 bus stops [71]. Spatial information, name, and identification of these public passenger bus stops are available [71]. These bus stops are used in the application environment of the proposed GPS and GIS model. Furthermore, spatial data of the rail tracks (train routes) in the city of Chicago is also available to the public [71], as geometric polylines. This rail line trajectory's location information is used in the proposed system. Spatial rail line information fused with GPS sensor data creates a classification feature in the proposed system. Transportation mode ground truth was labeled on the GPS sensor reports by each individual using the user interface (UI) of the mobile web application. In Table 1, we present a table, depicting the duration of mode specific labeled training data, collected from the travelers.

Table 1 - sensor data distribution



Mode	Time (min)
train	62
bus	69
car	52
walk	91
bike	47
stationary	53

4.5 GPS data preprocessing

The accuracy of GPS varies. For example, GPS tends to underperform if it does not have a clear view of the sky (e.g. in urban canyons). For this reason, we perform a noise filtering step before training the classifier. Invalid GPS points are suppressed based on the GPS accuracy (i.e. > 50m) and the change in speed. GPS sensor reports with high inaccuracy readings and unrealistic changes in speed are pruned. This is a manual step before classifier training. GPS noise filtering before classifier training is not a new concept. The authors of [11] also perform a preprocessing step before training their classifier.

4.6 Spatial indexing

Another concern is the performance of the system considering the fact that transportation network GIS data is used. Since GIS data is used for creating the classification features, the speed at which the classification features are created is important. If it takes a long time to create the classification features, a traveler may transfer from one mode of transportation to another undetected.

To enhance the computing efficiency, a spatial index is built on the geospatial knowledge. For one example, on the set of all bus stops, a spatial index is maintained to quickly determine the traveler’s spatial and temporal relationship with the bus stops.

We build a flat indexing scheme using zip codes; this scheme alleviates the overhead of doing the costly linear comparisons. First, we pre-compute the zip codes for all the bus stops, bus routes and train lines. Then, for each zip code in the city, we maintain a bus stop candidate list, bus route candidate list, and rail line candidate list. We cache this zip code index on the central server. Next, when a mobile user submits a GPS sensor report, instead of doing a linear comparison with all the bus stops, buses, and rail lines, we only compare against those in the

In this work, the zip code based pruning strategy is compared with other known spatial indexing strategies such as R-Trees [16] and GIST (Generalized Search Trees) [72] based indexing. Below, the evaluation tactic used to compare the different indexing schemes is discussed.

4.6.1 Spatial index performance evaluation

In order to evaluate the best spatial index to use in the proposed system, an evaluation of: (1) Zip code based pruning indexing strategy [8], (2) R-Tree based spatial indexing [16], (3) A combination of R-Trees [16] and GIST based indexing [72], and (4) Exhaustive scheme (a linear search for the desired spatial object) is conducted. The R-Tree based index [16] groups nearby geospatial objects and represents them with their minimum bounding rectangle in the next higher level of the tree. The GIST structure [72] divides the spatial objects into three sets: (1) objects to one side, (2) objects that overlap, and (3) objects that are inside. We also combine the GIST and R-Tree indexing strategies and measure the effectiveness.

More specifically, from over 10,000 GPS sensor reports that are submitted, the duration to create the classification features using the four approaches to spatial indexing (i.e. zip code pruning, R-Tree, R-Tree+GIST, and exhaustive) on the geospatial knowledge (e.g. bus stop locations) are compared. For geospatial knowledge, we used the entire transportation network for the city of Chicago. Chicago has the second largest transportation system in United States. The mean time to create the classification features across the 10,000 GPS sensor reports using the four spatial indexing approaches as shown in Figure 9.

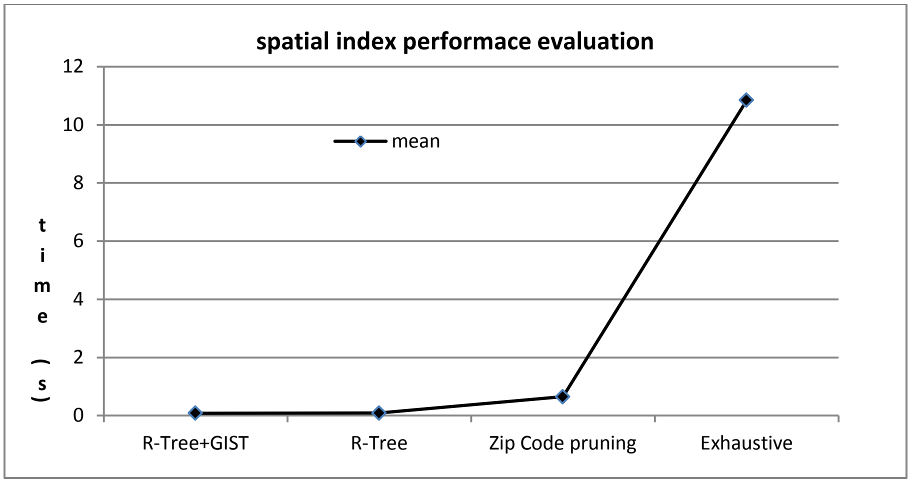


Figure 9-Spatial index performance

Table 2-Further performance statistics

	Std. dev	**Std. error**	**variance**
Exhaustive	0.70	0.02	0.49
R-Tree+GIST	0.02	0.0006	0.0004
R-Tree	0.01	0.0004	0.0001
Zip Code pruning	0.26	0.008	0.07

From Figure 9 it can be seen that the exhaustive scheme has the worst performance. The zip code based pruning strategy used in [8] is much better than the exhaustive approach; it takes 0.65 seconds to extract the mobility patterns, versus 10.84 seconds for the exhaustive scheme. The R-Tree [16] indexing strategy and GIST [72] combined with R-Tree [16] are the most efficient spatial indexing strategies in this work. The mean performance of both strategies is below 0.1 second. We also observed that a combination of R-Tree indexing and GIST based indexing does better than the R-Tree indexing by itself. In Table 2, we present further statistics such as standard deviation and standard error of the four geospatial indexing strategies.

4.7 Evaluation of GPS and GIS model

In this section, we analyze the performance and effectiveness of the transportation mode classifiers. We evaluate the classification schemes using two metrics: (1) *precision accuracy* and (2) *recall accuracy.*

Precision (M) = (*number of correctly classified instances of mode M) / (number of instances classified as mode M)*

Recall (M) = (*number of correctly classified instances of mode M) / (number of instances of mode M)*

The results obtained are presented in this section. Each set contains the precision accuracy and recall accuracy for the five classification schemes. For each set, we used 10-fold cross validation. In 10-fold cross validation, the original sample was randomly divided into 10 subsamples. Of the 10 subsamples, a single subsample was retained as the validation data for testing the classification model, the remaining 9 subsamples were used for training data. The cross-validation process was then repeated 10 times, with each of the 10 subsamples used exactly once as the validation data. The 10 results were then averaged to produce a single estimation.

4.7.1 Classification without transportation network

Figure 10 shows the first set of results which are the precision accuracy and recall accuracy when transportation network related features are not considered. The only features considered are average speed, average acceleration, average heading change, and average GPS position accuracy. Thus, real time bus locations, rail line trajectory and bus stop locations are removed.

From Figure 10 it can be observed that *Random Forest* classification model is the most accurate model since it has higher average precision and recall accuracy compared to the other four classification models. This Random Forest model is built from 10 decision trees, each decision trained with a subset of the classification features.

In general, when transportation network related features are not considered, the accuracy is below 76% for the five classification models (BN, NB, DT, RF, and ML). Additionally, we observe that motorized transportation and bikes show the lowest precision accuracy. For example, consider the case of Random Forest. The precision accuracy results for car, bus, and train are 58.1%, 56.5%, and 69.8%, respectively. The precision accuracy for bike is 71.4%. On the other hand, for non-motorized modes, such as walk and stationary, the precision accuracy is 100% and 96.8%. For all five models (BN, NB, DT,RF, and ML), the precision accuracy is best for walk and stationary.

	Precision					**Recall**				
	NB	**BN**	**DT**	**RF**	**MLP**	**NB**	**BN**	**DT**	**RF**	**MLP**
train	70.0	50.0	50.6	69.8	47.8	45.2	85.5	62.9	59.7	51.6
bus	47.0	43.9	40.6	56.5	37.8	54.4	31.6	22.8	45.6	24.6
stationary	100	100	100	100	96.2	100	100	100	1000	67.6
walk	94.7	93.8	93.8	92.7	83.3	97.8	100	98.9	97.8	98.9
car	42.3	90.0	43.5	58.1	30.2	42.3	17.3	38.5	69.2	25.0
bike	70.2	71.0	68.8	71.4	54.5	89.2	86.5	89.2	81.1	81.1
average	71.8	74.9	66.9	75.4	59.1	71.4	71.4	69.0	75.9	60.7

Figure 10-Transportation network features not considered.

This implies that the transportation network data is not very helpful for detecting stationary and walk mode. On the other hand, features such as speed, heading, acceleration, and GPS accuracy are not sufficient for distinguishing between motorized modes because of feature similarities.

4.7.2 Classification with Transportation network

Figure 11 shows the second set of results which are precision and recall accuracy when all the classification features discussed are used. The detection accuracy difference between Figures 10 and 11 is obvious. The main difference between Figure 10 and Figure 11 is that, in Figure 10, transportation network related classification features are not considered, while in Figure 11 transportation network related features are considered.

In Figure 11, all classifiers (**BN**, **NB**, **DT**, **RF**, **ML**) tested, with the exception of the Neural Network based Multilayer Perception (**ML**), achieve over 90% average precision and recall accuracy. On the other hand, when transportation network features are suppressed, the average precision and recall accuracy is below 76% (see Figure 10). This suggests that the transportation network related features are effective for transportation mode detection.

In the study, the most effective classification model is again Random Forest (**RF**), with an average precision accuracy of 93.7% and recall of 93.8%. This work is the first to distinguish between motorized transportation modes with such high accuracy [8]. From Figures 10 and 11, it can be seen that the precision and recall accuracy of motorized transportation modes and bikes increases more than the non-motorized modes of walk and stationary. For example, in the case of **RF** classification model, when transportation network features are used (Figure 11), the precision accuracies of car, train, bus, and bike are 87.5%, 98.4%, 88.3% and 88.9% respectively. On the other hand, when transportation network related features are not used (i.e. Figure 10), in the case of **RF**, the precision accuracies for car, train, bus, and bike are 58.1%, 69.8%, 56.5% and 71.4.% respectively.

We conclude that our novel transportation network features are most effective for motorized transportation mode detection, and also effective for bike mode detection. This makes sense, since bikes and motorized modes may have similar speed and acceleration in traffic and therefore are difficult to distinguish using traditional motion pattern features. However, features of the transportation network, such as bus locations, help the distinguishing between buses and bikes.

Distinguishing among motorized transportation modes is useful in practice. For example, companies such as Google collect data from travelers’ mobile phones in order to estimate the traffic speed of a road segment. For this purpose the speed estimation system should only use the speed reports submitted by mobile devices on cars or buses but not those on trains. Distinguishing the train mode from the other motorized modes enables the speed estimation system to filter out speed reports submitted from trains.

The proposed approach is 17 % more accurate than [14, 15] which uses GPS only and distinguishes between two motorized modes (bus and car). Also, the proposed approach is 9% more accurate than [12, 13] which uses GPS/GIS and distinguishes between bus and car. Using the newly proposed classification features, we show that we can detect transportation mode with high accuracy. These classification features are robust and most effective for detecting motorized transportation and bikes.

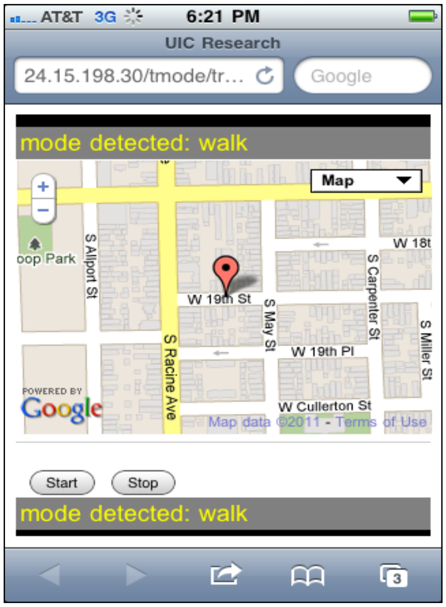
	Precision					**Recall**				
	NB	**BN**	**DT**	**RF**	**MLP**	**NB**	**BN**	**DT**	**RF**	**MLP**
Train	98.3	96.8	96.8	98.4	89.1	91.9	96.8	96.8	98.4	79.0
Bus	85.0	88.3	88.9	88.3	83.3	89.5	93.0	84.2	93.0	87.7
stationary	100	100	100	100	96.6	100	100	100	100	75.7
Walk	96.7	94.7	94.7	96.8	86.5	95.6	97.8	98.9	98.9	98.9
Car	78.2	85.4	85.1	87.5	67.3	82.7	78.8	76.9	80.8	67.3
Bike	88.9	88.6	85.5	88.9	75.0	86.5	83.8	94.6	86.4	73.0
average	91.6	92.5	92.2	93.7	83.3	91.4	92.6	92.33	93.8	83.0

Figure 11**-Transportation network features not considered.**

4.7.3 Extended real world evaluation of the deployed system

The final classification model (Random Forest), using the top five ranked features, was deployed to the public via the mobile web. As explained earlier, we focused on the centralized server model. In this model, mobile users submit their GPS sensor reports via the web to our central server for classification and activity inference.

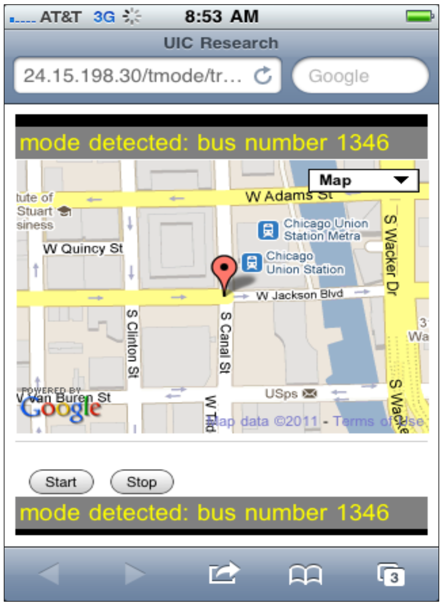
In Figure 12, we show the final deployed transportation mode classification system under operation in an IPhone 3G. When the detected transportation mode is “bus” in Figure 13, we provide further information by giving the bus’s identification. The bus identification is a finer granularity of transportation mode detection than bus route; we can also detect the bus route on demand. This work is the first to infer such detailed transportation mode detection. We also evaluated the deployed system to learn how the system performs in the real world under everyday usage.



mode detected: walk

mode detected: walk

Figure 12-Deployed classification system



mode detected: bus number 1346

mode detected: bus number 1346

Figure 13-Deployed classification system (bus mode)

For evaluation purposes, new individuals that were not considered for the initial training data collection were given IPhone 3Gs with access to the classification system. We considered new individuals for this experiment, because we wanted to learn how the system would perform for new users that are not covered by the training data. These new users mounted the IPhone in any desired position (i.e. waist, arm, pocket, or bag), then tally the percentage of time the mode detection is correct. For example, if the mode was detected 8 out of 10 times correctly, the accuracy is 80%. Table 3 shows that, when deployed in the real world, under everyday usage, we achieved an average detection accuracy of 93.42% for the proposed mode detection system. The results indicate that the proposed approach is effective under everyday usage, and new training data collection is not necessary for new users. Also, the identified transportation network related features are very robust to traffic condition changes.

Table 3-Evaluation of deployed system

Mode	Duration (min)	Accuracy %
train	35	93
bus	30	95
car	30	89
walk	30	92
bike	30	93
stationary	34	98.5

4.7.4 Discussion

In this section of the dissertation, we proposed a new robust approach to detecting transportation modes. In the proposed work, we considered and used transportation network data consisting of real time location of buses, rail lines, and bus stops spatial data. The real time location of buses is available in many cities such as Chicago, New York, Toronto, London, Washington DC, and San Francisco. Using the transportation network data, we showed that it is possible to address the weakness of previously proposed solutions [11, 12, 13, 14, 15]; that is, to distinguish between motorized modes such as trains, buses, and cars with high accuracy. Furthermore, if we detect that a traveler is traveling by bus, we can further identify on which particular bus the person is traveling.

Among the five classification models considered, Random Forest model is the most dominating classification model with over 93 % precision and recall accuracy. When transportation network classification features are not considered, the precision accuracy decreased to below 76%. This reduction of accuracy, upon omission of transportation network related features, is more notable for motorized transportation modes and bikes. This implies that transportation network data is effective for detecting motorized transportation, and bikes.

We also realized that, in order to achieve high precision and recall accuracy, only a subset of our initial set of classification features is necessary. In addition to traditional features on average speed and average acceleration, we identified for the first time the features on average bus closeness, average rail line closeness, and average candidate bus closeness. Using only this subset of features, and suppressing the other classification features that are not necessary, the precision accuracy was still over 92.5%.

Finally, users that have not participated in the initial training data collection evaluated the deployed system in the "real world" under everyday usage. The real world evaluation of the deployed system resulted in a precision accuracy of 93.42%. This indicates that additional training data collection is not necessary for new users; and the system is robust under every day usage.

4.7.5 Relevant work

The work of Zheng et al. [14, 15] is based on transportation mode detection from GPS data alone; the authors introduce a robust and novel set of machine learning features that are sensitive to certain traffic and weather scenarios. Our work is different in that we consider transportation network data such as the real time location of buses to build classification features. Additionally, [14, 15] do not consider train as a transport mode. The approach proposed in this thesis is over 17 % more accurate than [14, 15].

In [12, 13], the authors use an unsupervised learning technique to detect the transportation mode of a traveler. The transportation modes that are detected in [12, 13] include buses, cars and walk. Thework in [12, 13] is able to predict the traveler's goals, such as trip destination and trip purpose. In addition to GPS and GIS data, [12, 13] use historical information about the user. Historical information includes, past user trips and information about where the users parked their cars. In our approach we do not consider historical information about the user. Furthermore, we use a supervised learning mechanism to detect transportation modes from the set {WALK, BUS, DRIVING, TRAIN, STATIONARY, BIKE}. Another difference is that we use different transportation network data than [12, 13] do. Particularly we use real time bus locations, rail line spatial data, and bus stop spatial data. [12, 13] use historical information about the bus stops at which a user boards, and where the user parks her/his vehicle. Importantly, the proposed bus stop feature is different than that in [12, 13]; the proposed classification feature captures the number of bus stops and duration at bus stops. A weakness of [12,13] is that the users' motion pattern such as where the user parks her/his vehicle daily are taken into consideration, and therefore the model relies on background information about the user. The accuracy of the proposed approach is higher than that of [12, 13] by 9%.

The proposed approach uses a single sensor (i.e. GPS) on the mobile device. There have been studies that consider multiple sensors for transportation mode recognition [31, 32, 35, 36,]. In [35, 35], over 20 sensors that are wearable on the human body are used. The input to the classification model includes information on the user's body condition such as temperature, heart rate and GPS position. We consider a smaller number of sensors, but add transportation network data. We believe that it is unlikely for normal users to carry over 20 sensors daily. The work in [31] utilizes multiple accelerometers and [32] uses a single sensing unit with multiple sensors (accelerometer, audio, and barometer) for activity detection. The state of the art is [11] which uses GPS and accelerometer sensors for transportation mode detection. However, [11] does not distinguish between motorized transportation modes such as car and bus. This limitation is due to the similarity in features of these two modes of transportation, especially in traffic or extreme weather. Using GPS and GIS data, as shown in the proposed approach, can achieve a very high detection accuracy, as in [11]. However, in the proposed approach we distinguish between motorized transportation modes and we do not use accelerometer as in [11]. The work in [34] is purely based on GSM. In [73], the only sensor considered is the triaxial accelerometer. In [25], the authors' objective is to conserve mobile devices resources. Table 4 summarizes the related works that uses GPS.

Table 4- Relate work with GPS sensor

	Classes	**Sensor**	**Time**	**Users**	**Accuracy**
[2]	driving, bus, bike, walk	GPS	10mths	65	76.2%
[4]	still, walk, run, bike, motor	GPS, accelerometer	50days	16	93.6%
[1,16]	walk, bus, car	GPS,GIS	60days	1	84%
[15]	car, bus, bike, walk	GPS	6 mths	45	74%

5. INDOOR AND OUTDOOR ACTIVITY DETECTION

The mobile industry is witnessing a rise in context aware mobile services that collect sensor data from mobile phone users, analyze these data sets to identify higher order attributes, and then feed these abstractions back to serve the users. This gives rise to sensor data driven service innovation, where mobile service ideas are generated based on patterns related to individuals as well as social networks. The key idea is to create compelling user experiences based on the patterns extracted from the sensors on the traveler’s mobile phone. Mobile phones now have increased computational, sensing, and communication capabilities. With these improved capabilities of mobile phones, several applications have been implemented. These applications include, but are not limited to: sensor sharing of information in social networks (e.g. location sharing), every day activity recognition (e.g. context detection), and mobile peer to peer applications (e.g. photo sharing).

In the prior section (i.e. Section 4), the dissertation discussed outdoor transportation mode detection using GPS and transportation network data. Some applications such as calorie trackers [26, 27, 28] that compute the number of calories that a traveler consumes need a finer granularity of transportation mode detection than solely outdoor modes. For example, they should be able to detect indoor transportation modes such as stairs, elevators, and escalators since they all require different amount of energy or calorie to utilize and would enable a more accurate reflection of the traveler’s activities.

In this section, we focus on both indoor and outdoor transportation mode detection using the GPS and accelerometer on the traveler’s mobile phone. These two sensors are common features on today’s mobile phones. In addition to GPS and accelerometer on the traveler’s mobile phone, we also utilize geospatial knowledge of the underlying transportation network. As discussed, geospatial knowledge of the underlying transportation network includes real time bus locations, bus stop locations, and rail line spatial information.

Given the accelerometer and GPS sensors on one's mobile phone, the proposed work combines information from these two sensors with transportation network data to create mobility patterns that are unique and can identify different modes of transportation. The transportation modes identified in this work include the following:

- Stairs
- Escalator
- Elevator
- Car
- Bus
- Aboveground train
- Walk
- Stationary

Stairs, escalators, and elevators are referred to as indoor transportation modes. Car, bus, aboveground train, walk, and stationary are referred to as outdoor transportation modes.

To identify the transportation mode of a traveler, we follow the general principle of supervised learning from mobile sensor traces [8, 11]. First, labeled sensor (i.e. GPS and accelerometer) data is collected from the traveler's mobile phone. The initial data labeling is done by the travelers from their mobile phones where they annotate the transportation mode that corresponds to their trip and sensor reports. This labeled sensor data is then fused with GIS information about the transportation network to create classification features. For example, using the GPS sensor reports, spatial properties with the closest bus, bus stop, or rail line can be computed. For another example, from the accelerometer sensor, the accelerations in the x, y, and z planes are used as features. With these classification features, a machine learning model is trained with the labeled sensor data and GIS information combined. Then, when the transportation mode is to be determined, unlabeled sensor data (i.e. GPS and accelerometer data) from the traveler's mobile phone is fused with GIS information to create the same features that were used for training. These features are then fed to the already trained classification model. This trained classification model can then determine the current mode of transportation used by the traveler in a probabilistic format in real time. We were interested in quantifying the following: (1) Which of the classification models is most effective when accelerometer, GPS, and GIS data is considered, (2) Whether combining several base level classifiers to form meta-level classifiers is most effective for transportation mode detection in this work, (3) From the selected feature set, which features are most important, and (4) Which transportation mode activities are more difficult to recognize.

5.2 Sensor selection and transportation network

There are several sensors on today's mobile phones. Such sensors include microphone, camera, GPS, gyroscope, accelerometer, and Bluetooth. In this section, we will discuss the sensors used in this work, sensor reports, and transportation network that are utilized in the proposed work.

5.2.1 Sensors

There are several sensors on today's mobile phone. Sensors include GPS, accelerometer, microphone, camera/s, gyroscope, etc. Data streams from two sensors on the traveler's mobile phone are used to detect the transportation mode that a traveler is using. The two sensors are: (1) Accelerometer, and (2) GPS. These two sensors are commonly available and accessible on today's mobile phones.

5.2.1.1 Accelerometer sensor

The accelerometer measures acceleration in all three axes (i.e. x, y, and z). This sensor has been used extensively in activity recognition [11, 29, 30, 31, 35, 36, 74]. From a mobile phone, one can access the acceleration with gravity and also without gravity and can then compute statistical features that are unique for a particular activity. Using these statistical features such as mean, root mean square, and sum of area encompassed by the magnitude of the readings on the three axes (i.e. x, y, and z), one can build a threshold model or a classification model to detect or infer the user's current context probabilistically. Several papers consider frequency domain techniques such as Fourier Transform (example Fast Fourier Transform) and Fast Time Frequency Transform for context detection. FFT enables the frequency domain representation of a time based signal. The accelerometer is useful in distinguishing between motorized and non-motorized transportation modes as observed in [11]. However, to distinguish between different motorized modes such as car and bus using only accelerometer is challenging since the force vector values for motorized transportation modes are similar. Please see Chapter 3 and definition 3 of this dissertation for the formal definition of an accelerometer sensor report.

5.2.1.2 GPS sensor

The GPS sensor can provide accurate location positioning data. From GPS, properties such as latitude, longitude, speed, acceleration, and heading change can be computed. Given these attributes, it is possible to infer a person's context [8, 11, 12, 13, 14, 15]. For indoor context detection, GPS may not be suitable since accuracy decreases indoors. Another pitfall of using GPS for context detection is its high power consumption. Consequently, the sampling period is important for context detection when using GPS sensors. The sampling period of the sensors and energy conservation is explored in this chapter. Please see Chapter 3 and definition 1 of this dissertation for the formal definition of a GPS sensor report.

5.2.2 Transportation Network

Transportation network considered in this chapter is described in Section 3.4 of this dissertation.

5.3 General algorithm

The general algorithm for transportation mode detection has three stages: (1) Feature selection from mobile phone sensor trace, (2) Model training, and (3) Transportation mode inference.

In stage 1, the GPS and accelerometer data from the traveler's mobile phone is combined with geographic information of the transportation network. Given these three pieces of information (i.e. accelerometer, GPS, and GIS), classification features that are useful for each mode of transportation are created. The procedure is illustrated in Figure 14. These classification features will be discussed thoroughly in subsequent sections of the thesis. In this stage, a noise filtering step on the GPS sensor reports is performed, since inaccurate GPS reports can mislead the system. The filtering is based on the accuracy parameter of the GPS sensor report. GPS sensor reports with accuracy readings greater than 50m radius are discarded. The GIS data maintains a spatial index for performance reasons.

In stage 2, given the labeled classification features from stage 1 for all the modes of transportation, multiple classification models are trained. These models include base-level models such as Decision Trees and more advanced complex meta-level classification models such as Voting [75] and Stacking [76].

Finally, in stage 3 (inference stage), to determine the traveler's transportation mode, the same features utilized in stage 1 are extracted. Subsequently, given these features, the already trained classification system predicts the transportation mode of the traveler, probabilistically.

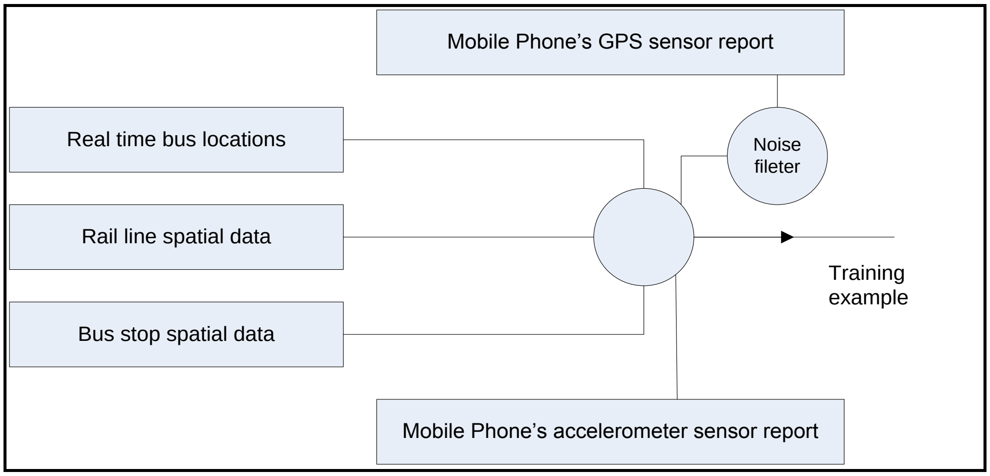


Figure 14- **Classification examples**

5.4 Classification feature extraction

This section details the classification features that are used for transportation mode detection in the proposed work. The intuition behind the features is that transportation mode traveling patterns will be reflected in the sensor traces. In [8], several new classification features were introduced by combining the GPS sensor trace with geographic knowledge of the transportation network. This work will extend that classification feature set used in [8] by adding the new features described below. Thus, we will use the GPS and GIS features from our prior work [8] in conjunction with new accelerometer features, for indoor and outdoor transportation mode detection on mobile phones.

In Figure 15, we show raw accelerometer readings for various indoor and outdoor transportation modes. The readings for all three axes are shown.

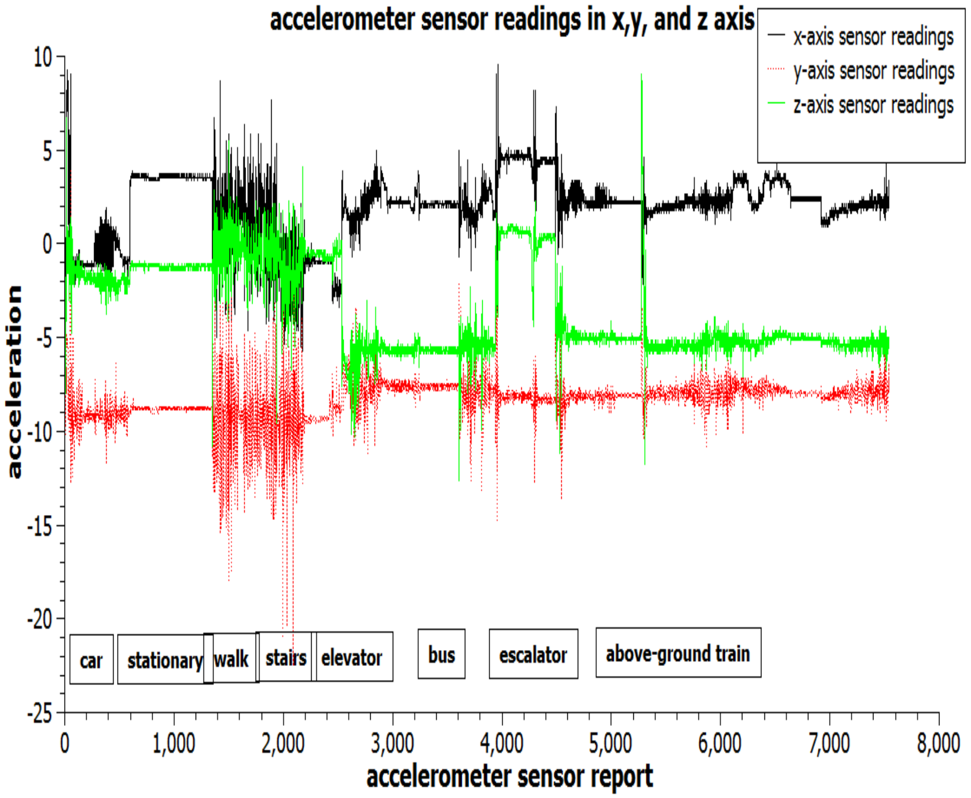


Figure 15- **Accelerometer raw data sampled at 50 times/second. Vertical dimension is y.**

In Chapter 4 the following features were used based on GPS and GIS: (1) speed, (2) acceleration, (3) average bus closeness, (4) candidate bus closeness, (5) rail line closeness, (6) bus stop closeness rate, (7) heading change, and (8) GPS accuracy. These features are effective for detecting outdoor transportation mode and distinguishing between motorized and non-motorized transportation modes as highlighted by [8].

In this chapter, based on the addition of new sensors such as accelerometer to the GPS/GIS model, the following classification features are added:

Mean of acceleration: The mean acceleration in the x, y, and z axis should vary across transportation modes. For indoor transportation modes such as elevators, the vertical accelerometer reading should be prevalent since elevators travel vertically. On the other hand, for outdoor modes such as train, bus, and car, the horizontal accelerometer readings are prevalent. Apart from the mean in the individual axis (i.e. x, y, and z), the mean across the three axes was also used as a feature. Let $\{a_1, a_2, a_3, a_4 \ldots a_n\}$ be a finite set of accelerometer sensor reports submitted within a time window ***t*.** We compute the mean acceleration in all three axes as:

$$mean_{xyz} = \frac{(\sum_{i=1}^{n}(|x(i)|) + (|y(i)|) + (|z(i)|))}{n}$$

where $x(i)$, $y(i)$, and $z(i)$ represent the acceleration component of sensor report a_i along the x, y, and z axis respectively obtained from the accelerometer.

Standard deviation of acceleration: Standard deviation in the x, y, and z axis is used to capture the fact that the range of possible acceleration values differ between transportation mode activities.

yx ratio of acceleration: Acceleration in the y axis can help identify transportation activities that have vertical movement components such as riding elevators, escalators or climbing/descending stairs. However, the ratio of yx is different for all these transportation modes. For example, even though elevators have y axis acceleration, they have limited x axis acceleration. Likewise, escalators have y axis acceleration and also x axis acceleration.

High Peak and low peaks of acceleration: High and low peak acceleration in the x, y, and z axes from a set of accelerometer sensor reports. Unlike the mean acceleration, the high and low peak acceleration for a set of sensor reports can detect sudden brisk movements. These features are important to distinguish between indoor modes and outdoor modes, such as climbing stairs, elevator or escalator from bus or car. For example, for y peak acceleration, we can distinguish indoor activities from outdoor activities since the indoor activities have vertical movements. More specifically, stair climbing activity can produce higher y axis acceleration peak values than elevator or escalator. Furthermore, z peak acceleration can capture sideward activities such as the bus pulling over to a bus stop.

Correlation of acceleration: Correlation measures the dependency between two variables. In this work, the correlation is calculated between each pair of accelerometer axis (i.e. xy, xz, and yz) as the ratio of the covariance and the product of the standard deviations. Thus, for example:

$$corr\ (x,z) = \frac{cov(x,z)}{\sigma_x \sigma_z} = \frac{E[(X-\mu_x)(Z-\mu_z)]}{\sigma_x \sigma_z}$$

where $\mu_x\ and\ \mu_z$ are the expected values and $\sigma_x \sigma_z$ are the standard deviations. Correlation is useful for distinguishing activities that involve translation in multiple dimensions. For example, using correlation, we can distinguish between driving and climbing stairs. Climbing stairs translates in multiple dimensions. Let us consider for another example the feature corr (y, z); the correlation between the y and z axes. For transportation modes such as stairs and walk, the corr (y, z) will have low values since the standard deviation for acceleration in the y and z axes is high. On the other hand, elevators have high corr (y, z) since it has smaller deviations and more profound vertical movements.

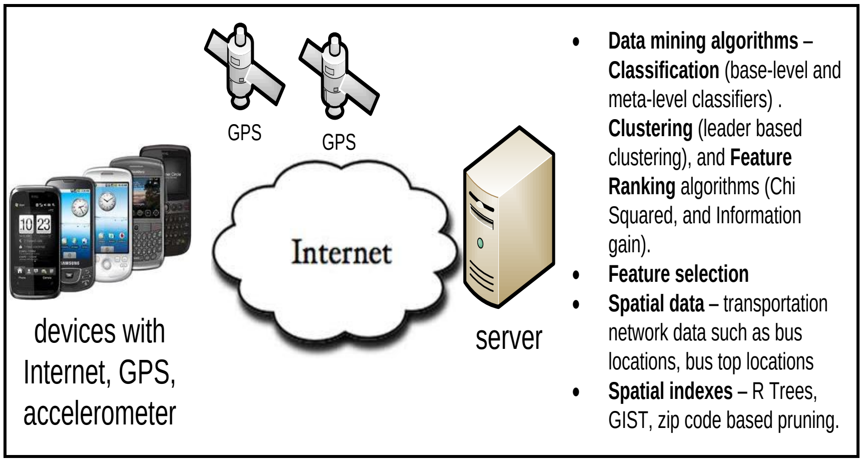
Signal magnitude area (SMA) of acceleration: SMA has been used in previous work on user activity recognition to distinguish between resting states and other user activities [77]. For SMA, the sum of the area encompassed by all three signals is used to compute the energy expenditure. Using the acceleration due to body movements, the total value of the accelerometer signal can be computed as:

$$SMA = \frac{1}{t}(\int_0^t |x(t)|dt + \int_0^t |y(t)|dt + \int_0^t |z(t)|dt)$$

where *x(t)*, *y(t)*, and *z(z)* represent the acceleration component along the x, y, and z axes respectively with respect to time. Since the SMA is the sum of the area encompassed by the three axes, transportation mode activities such as walking have higher SMA than stationary.

5.5 Architecture for indoor and outdoor transportation mode detection

A centralize system architecture (see Figure 16) is utilized in this work. A traveler's mobile device has a GPS, accelerometer sensor, and an Internet connection. The mobile device submits GPS and accelerometer sensor reports to the central server automatically via the Internet with the traveler's permission. After the central server receives the sensor reports, it identifies the transportation mode that is being used.



- **Data mining algorithms – Classification** (base-level and meta-level classifiers) . **Clustering** (leader based clustering), and **Feature Ranking** algorithms (Chi Squared, and Information gain).
- **Feature selection**
- **Spatial data** – transportation network data such as bus locations, bus top locations
- **Spatial indexes –** R Trees, GIST, zip code based pruning.

Figure 16- System architecture (centralized model)

Let us compare the centralized model with the distributed model where activities are monitored locally on each mobile device. In the centralized model, the transportation mode detection activities are monitored at the central server, so there is no need to store spatial data on the mobile phone. Hence, the centralized model consumes less of the device's memory. On the other hand, the distributed model is location privacy aware, since the user's location is not submitted to a central authority. Privacy in mobile systems with the centralized architecture is addressed in prior work [6, 7].

5.6 Mobile device position

One concern is that the placement of a mobile phone on the traveler might affect the quality of the sensor data received. For example, the accelerometer on a person's mobile phone may produce different signals if the mobile phone is placed in a bag or in one's pocket. In [8], device placement was not evaluated since only the GPS sensor on the traveler's mobile phone is used. The mobile phone's GPS sensor is more withstands body movements or body placement better than accelerometer in terms of the signals it produces. In this work, since the accelerometer is utilized, it makes sense to determine a unique position for the device to produce accelerometer sensor readings consistently. This work is not the first to identify issues of accelerometer sensor signal variations with device orientation. Issues with device positioning and accelerometer readings have been documented in the literature [32, 78]. For one example, the work in [37] requires the experimental participants to wear an elastic belt at their waist with the devices placed above the right iliac crest for the entire monitoring period.

To answer this question of device position, a mobile phone is placed in three different positions on a traveler. The traveler then walks on a level surface, and finally goes up and down a stair case. Figures 17-25 show the accelerometer signal from an activity from three different mounting positions: firmly attached to the travelers side in a case, in the traveler's front pocket, and in the traveler's bag. The accelerometer sensor reading was observed in all three axes (i.e. x, y, and z). The first position (i.e. bag) was tested because many females utilize this method for carrying a mobile phone. The anomalies at the first and end part of the signals are from the device being mounted and removed from its desired body position.

For the case of the mobile phone in front pocket, we observed that, many unwanted signals are produced. This is due to the bumping of the mobile device by the traveler's leg as she walks. For the case of the mobile device being placed in the traveler's bag, unpredictable results are produced. Consequently, in our experimental evaluation we attached the mobile phone to the participants' waist/side using a phone case.

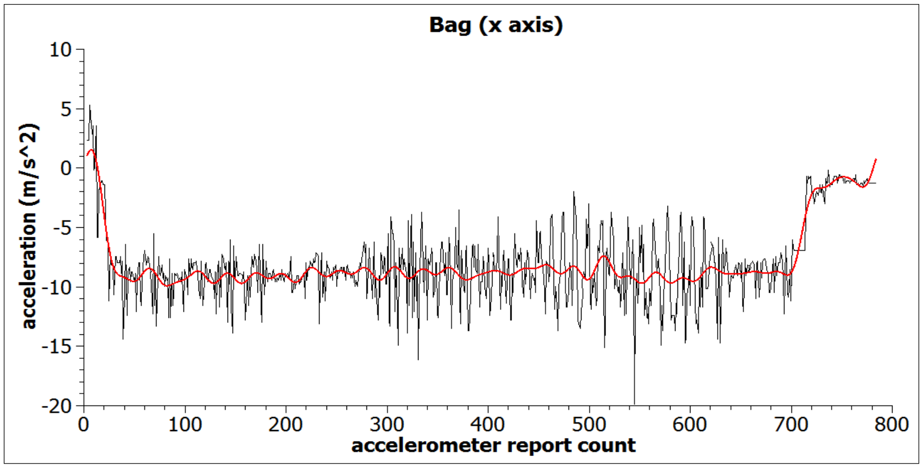


Figure 17- mobile device in bag (x axis readings)

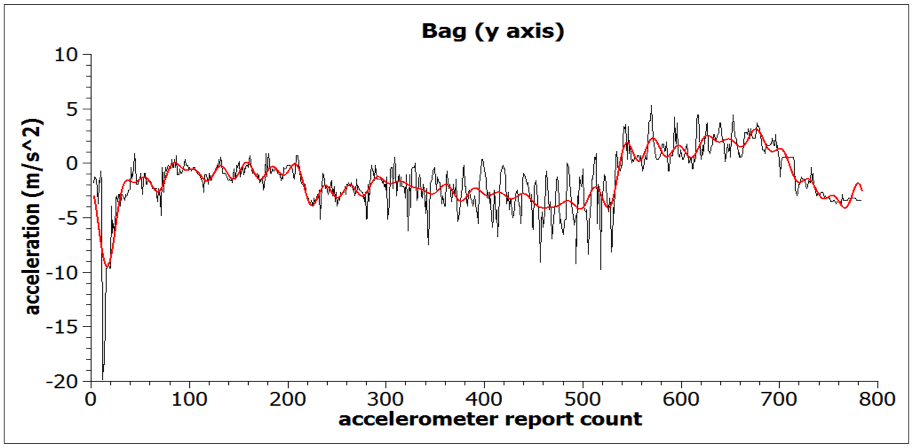


Figure 18 - mobile device in bag (y axis readings)

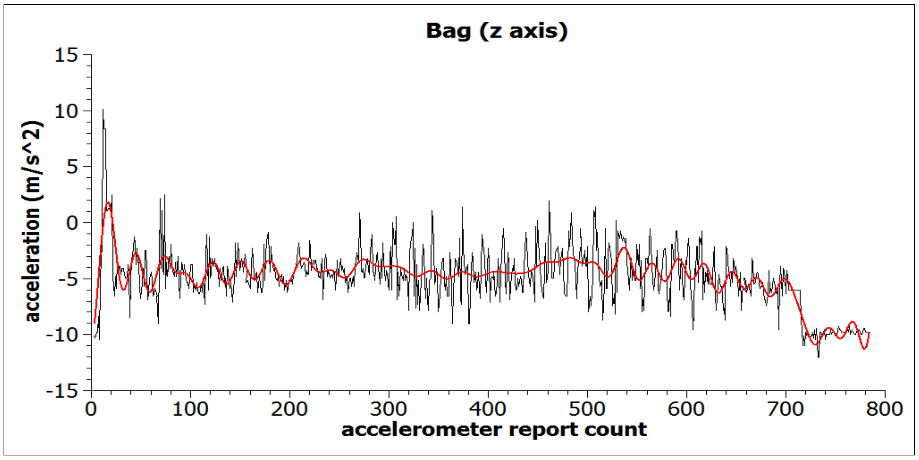


Figure 19-mobile device in bag (z axis readings)

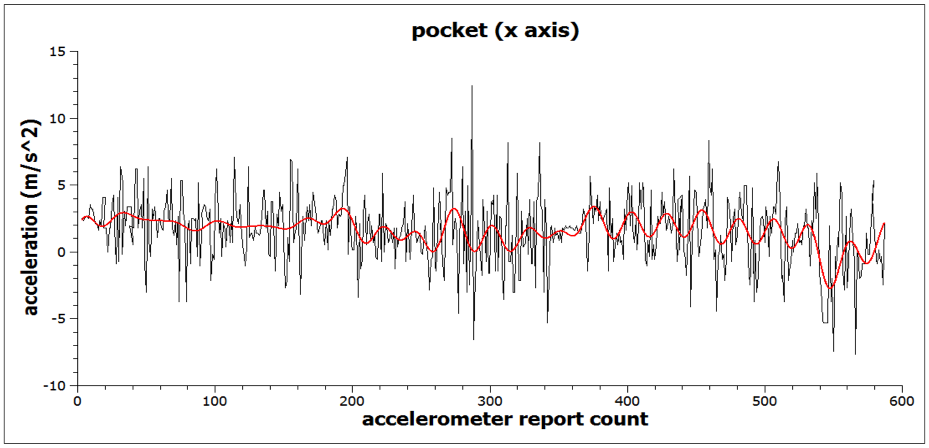


Figure 20-mobile device in front pocket (x axis readings)

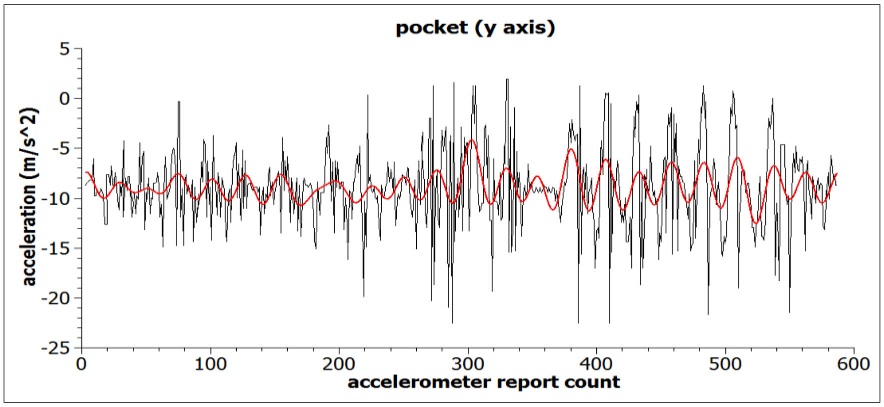


Figure 21-mobile device in front pocket (y axis readings)

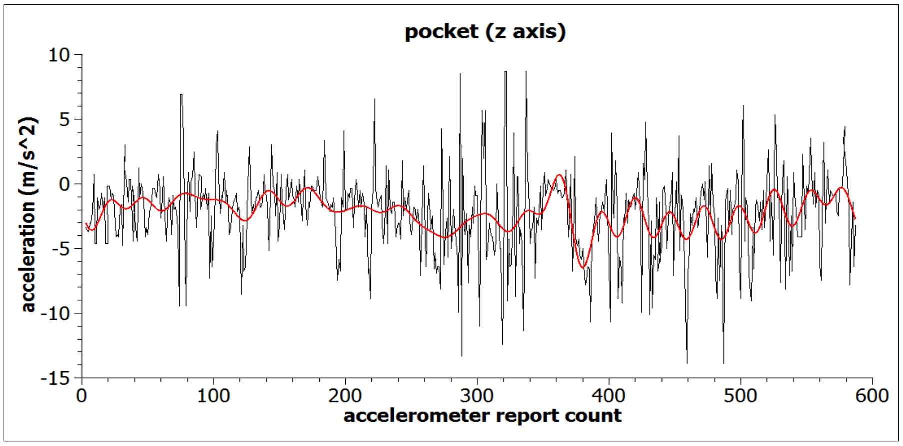


Figure 22-mobile device in front pocket (z axis readings)

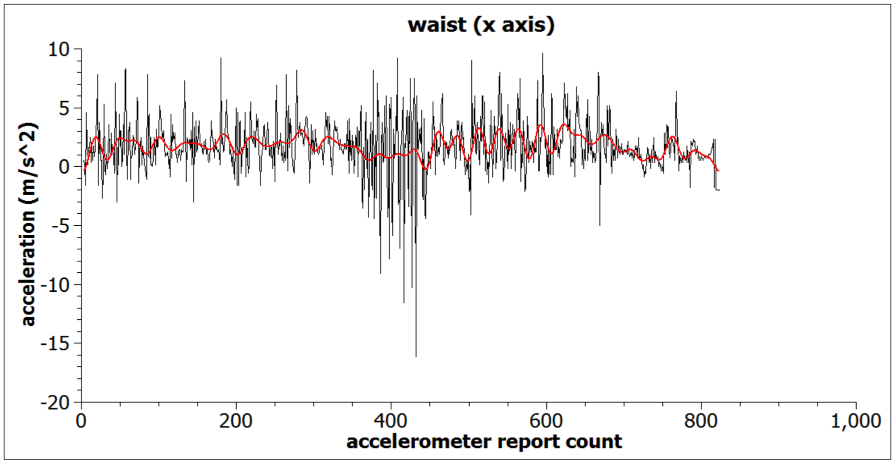


Figure 23-mobile device on waist (x axis readings)

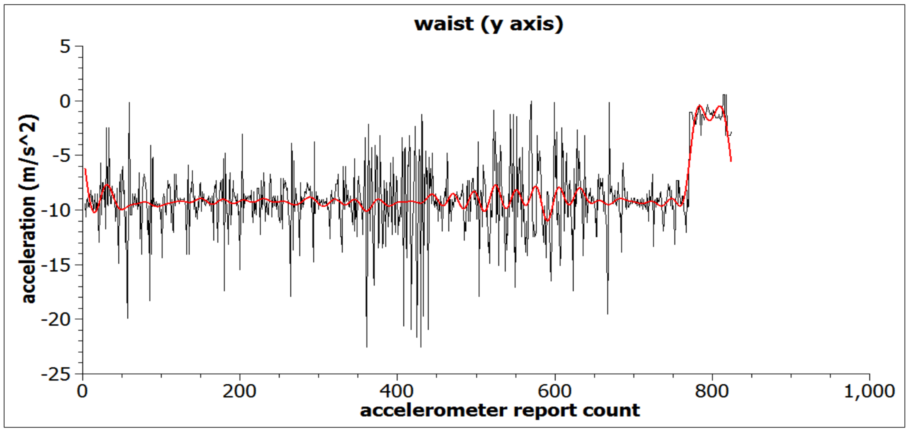


Figure 24-mobile device on waist (y axis readings)

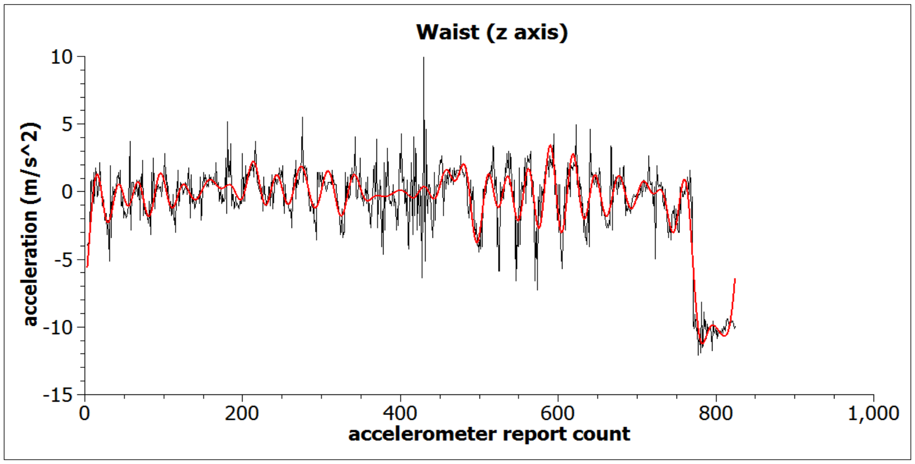


Figure 25 – mobile device on waist (z axis readings)

5.7 Evaluation by experiments

In this section, the collection of mobile sensor data, classifier selection, experimental setup and experimental results are deliberated.

5.7.1 Experimental setup and sensor data collection

Two types of mobile phones are considered for the experiments (1) IPhone 4S and (2) iPhone 3G. Constrained experimental setup was considered where experiment participants were instructed to place the mobile phone in a phone case vertically on their sides. This ensured that the collected data is uniform and unwanted noise was not introduced to the data. The devices were unlocked for the duration of data collection to mitigate premature accelerometer deactivation when the iPhone is locked.

For training the classifier, traces on eight different modes of transportation (walking, bus, car, stationary, aboveground train, stairs, elevator, and escalator) is collected. The data were collected by 4 individuals over a 2 week period and represents over 3.47 hours of sensor data in total. One limitation is the size of the dataset considered. Realistically, we need to collect more sensor data (GPS and accelerometer) in an uncontrolled manner from several travelers to effectively study and validate the proposed model. The dataset considered provided an insight on indoor and outdoor transportation mode activity classification, however with more sensor data including noise, the classification accuracy may be reduced. For larger scale experiment, one may consider switching to Android phone which allows background applications to collect sensor data. Such that the subjects can still use their phones as it is and generate more realistic data in the wild. Leaving the phone unlocked with data collection application running on the foreground is for a proof of concept, but not the best option for a longer term study.

See Figure 26 for the distribution of transportation modes in the collected data. Our application platform is the mobile web, and the transportation mode ground truth is labeled on the sensor reports by each experiment participant via the User Interface (UI) of the sensor data collecting application.

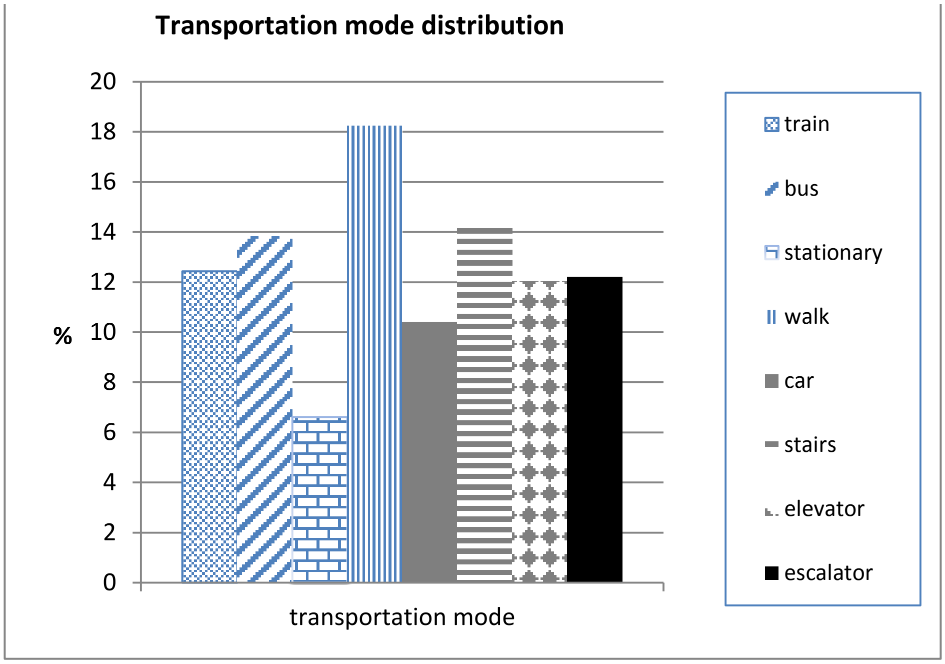


Figure 26-Data distribution

5.7.2 Classifier Selection

The classifiers that are used in the proposed work are discussed in this section. More specifically, this work will re-use some of the classifiers used in [8] in addition to new classifiers. In [8], all the classifiers used are base-level classifiers. In this work, in addition to the base-level classifiers used in [8], more advance classifiers called meta-classifiers are utilized. Meta-classifiers can combine several base-level classifiers in order to improve prediction accuracy. To evaluate the different classification models, the WEKA machine learning tool set [79] was considered.

5.7.2.1 Base level classifiers

In this work, several base-level classifiers are used to measure the transportation mode detection accuracy. The base-level classifiers considered are Random Forest and Decision Trees. These base-level classifiers were used in [8]. For literature on these base-level classifiers, we encourage readers to see [79, 80]. The transportation mode detection accuracy of these simpler classifiers compare to the more complex meta-level classifiers described below is investigated.

5.7.2.2 Meta-level classifiers

Boosting [81] – Given a base-level classifier, one can use boosting to improve the classification accuracy. Boosting works by applying a single learning algorithm repeatedly then combines the hypothesis that is learned on each occasion. Boosting assigns a weight to each example in the training data, then it modifies the weight after each iteration depending on whether the example is classified correctly or not. The final hypothesis is:

$$f\,(x) = \sum_{t=1}^{T} \alpha_t h_t\,(x)$$

where $h_1, h_2 \ldots h_t$ is the set of hypothesis and α_t is the coefficient with which the ensemble member h_t is combined. Both α_t and h_t are learned during boosting.

Bagging [76] – Bagging is a simple meta-level classifier that uses one base-level classification strategy at a time. Given training data, the bagging scheme trains each individual classifier on a random distribution of the given training data. This method uses multiple versions of a training set by using bootstrap, i.e. sampling with replacement. The final bagged estimator $h_{bag}\,(.)$ is the expected prediction across each trained hypothesis. Where M is the M instances from the training set and $h_p\,(.)$ is the learned hypothesis for training sample p.

$$h_{bag}\,(.) = \frac{1}{M} \sum_{p=1}^{M} h_p\,(.)$$

Plurality Voting [75] – Voting selects the class that has been predicted by majority of the base classifiers as the final detected class. For the case where class probability distributions are predicted by base-level classifiers, there is a refinement of the voting algorithm. In this case, the probability distributions obtained from the base-level classifiers are aggregated to produce the class probability distribution of the meta-level classifier.

$$P_{ML} = \frac{1}{|C|} \sum_{c \in C} P_c\,(x)$$

Stacking [82] - In stacking, the predictions from different classifiers are used as inputs into meta-level classifiers, it then attempts to combine the predictions to create the best possible predicted classification. For example, the predicted classifications from the tree classifiers, Bayes Net model, and the Neural Network classifiers can be used as input variables into a Neural Network meta-level classifier, which will attempt to "learn", from the data, how to combine the predictions from the different models to yield maximum classification accuracy.

5.8 Results

This result of this work is organized as follows. First, we will present results on indoor transportation mode detection, using the set of features in [8] along with the new classification features that are based on the accelerometer sensor. Then, we present results on outdoor transportation mode detection. Finally, results on the combination of both indoor and outdoor transportation mode detection are presented. For each study, we used two base-level classifiers (1) Decision Trees (**DT**) [79, 80], and (2) Random Forest (**RF**) [79, 80] and several meta-level classification strategies such as bagging [76], boosting [81], stacking [82], and voting [75]. We choose the base-level methods based on previous studies. For example, in [11], it was pointed out that a Decision Tree model was the most effective base-level classification scheme for outdoor transportation mode detection, when GPS and accelerometer data is utilized. For another example, in [8] it was pointed out that a Random Forest model was the most effective classification scheme, when GPS and GIS data is utilized. Thus, we wanted to quantify the effectiveness of these two base-level classifiers (i.e. Decision trees and Random Forest), along with more elegant meta-level classifiers. Ten-fold cross validation was conducted to get the performance of each classifier. For each classifier, we will discuss the detection accuracy in terms of F Score. The F Score is the harmonic mean of the precision and recall accuracy.

5.8.1 Indoor transportation mode detection

The indoor transportation mode (i.e. stairs (**ST**), elevators (**EL**), and escalators (**ES**)) detection accuracy is studied in this section. Indoor transportation mode detection is useful, for example, to infer the presence of an elevator or escalator at a location, or to automatically determine whether travelers are using the elevators, escalators, or stairs. We will present the F Score for each indoor transportation mode. Additionally, a discussion on the ranking of classification features in order of their effectiveness in predicting indoor transportation modes is presented.

Table 5- F Score of indoor transportation mode detection (GPS/GIS and accelerometer model)

	F-Score			
	ST	EL	ES	Average
DT	98.6	99.2	97.6	98.5
Bagged DT	98.6	99.2	97.6	98.5
Boosted DT	98.6	99.2	97.6	98.5
RF	98.6	100	98.4	99
Bagged RF	98.6	99.2	99.2	99
Boosted RF	98.6	99.2	99.2	99
Stacked	96.4	100	96.0	97.5
Vote	99.3	100	99.2	99.5

The results of Table 5 indicate that the proposed work can effectively detect indoor transportation mode. Both of the base-level classifiers (i.e. Decision trees (**DT**) and Random Forest (**RF**)) achieved over 97% accuracy. It is observed that combining several base-level classifiers to form meta-level classifiers isn't much more effective than using only base-level classifiers for indoor transportation mode detection from GPS, accelerometer, and GIS data (i.e. transportation network). For stacking and voting meta-classifier, we combined several base-level classification strategies. The strategies are Decision Trees (**DT**), Random Forest (RF), Naïve Bayesian, and the neural Networks based Multi-layer Perceptron.

In general, for elevators, the dominant vertical movement makes it easily distinguishable from climbing stairs and riding escalator. For stairs, the peaks from the accelerometer sensor, among other features, make it distinguishable.

In addition to determining the accuracy with which the proposed work can detect that a traveler is using a particular indoor transportation mode, we wanted to know which sensor (i.e. GPS or accelerometer) is more important for indoor transportation mode detection. In order to determine the most effective sensor, we used previously proposed classification feature selection schemes to rank the classification features that are used to detect indoor transportation modes. Feature selection is a data-mining concept which chooses the subset of input features by eliminating classification features that are less predictive. For feature selection, we used the Information Gain feature selection algorithm [83].

Using the Information Gain [83] feature selection algorithm, the top five most effective classification features for indoor transportation mode detection are: (1) corr (x, z), (2) mean of accelerometer x axis signal, (3) low peak of y axis signal, (4) yx ratio, and (5) high peak of the y axis signal. On the contrary, the five least effective classification features for indoor transportation mode detection are: (1) average bus closeness, (2) acceleration computed from GPS, (3) bus stop closeness rate, (4) accuracy of GPS signal, and (5) heading change computed from GPS signal. Consequently, the Information Gain ranking algorithm provided some amount of evidence that the accelerometer sensor is more effective for indoor transportation mode detection than the GPS/GIS model, since the accelerometer classification features are more effective.

Even though the Information Gain feature ranking algorithm provided some evidence that the accelerometer related classification features are more effective than those of GPS and GIS for indoor transportation mode detection. We wanted to further quantify this analysis by supplementary experiments. For quantification, we studied the indoor transportation mode detection accuracy when only the accelerometer is used and also when only the GPS/GIS is used.

When only the accelerometer sensor is used for indoor transportation mode detection, the F Score is over 95% across indoor modes. On the other hand, when only GPS with GIS is utilized for indoor transportation mode detection, the F Score goes below 67%. In Tables 6 and 7, the confusion matrices for both cases (i.e. only accelerometer sensor and only GPS with GIS) are depicted.

Table 6-Confusion matrix for accelerometer sensor only

	Stairs	Elevator	Escalator
Stairs	69	0	2
Elevator	0	60	0
Escalator	0	1	60

Table 7-Confusion matric for GPS/GIS model

	Stairs	Elevator	Escalator
Stairs	47	14	10
Elevator	16	37	7
Escalator	10	8	43

From the confusion matrixes in Tables 6 and 7, there are several misclassifications among indoor transportation modes when only the GPS/GIS is utilized (see Table 6). In other words, from Table 7, the classification system constructed with only GPS/GIS features is less accurate than that of accelerometer only (Table 6). This makes sense since when one is traveling using an indoor mode, their GPS location is not deviating much. When only the accelerometer sensor is used for indoor transportation mode detection, the detection accuracy is 28% higher as discussed, and there is less misclassification among modes (see Table 6). Additionally, apart from the classification accuracies and misclassifications observed from the confusion matrix, the Information Gain method had also provided other evidence on sensor effectiveness by ranking the accelerometer-based classification features higher than the GPS/GIS related features. Consequently, we claim that for indoor transportation mode, the accelerometer sensor is more effective than the GPS/GIS model.

5.8.1.1 Elevator detection discussion

In collecting the sensor reports for training the classification system while a traveler is riding an elevator, we made sure that the traveler has an uninterrupted Internet connection in the elevator. Consequently, in this work, we first test elevators for un-intermittent Internet connection. Since a centralized system is used, we wanted to make sure we receive all the sensor reports that are submitted. Alternatively, one could cache the sensor reports on the mobile phone and submit the sensor reports when an Internet connection becomes available. However, we were interested in creating a system that is useful in real time.

In this work, high accuracy for elevator mode is achieved. Elevator transportation mode is unique in the sense that the vertical movements are predominant. Also, the vibrations of the elevators as it stops and proceeds at each floor are distinctive. In the case of escalator and stairs accelerometer signals, these patterns are not present.

5.8.2 Outdoor transportation mode detection

The outdoor transportation mode (i.e. train (**T**), bus (**B**), stationary (**S**), walk (**W**), and car (**C**)) detection accuracy is studied using the GPS, accelerometer, and transportation network in this section. We will present F Score for each outdoor transportation mode.

The first observation from Table 8 is that the outdoor transportation mode detection accuracy is higher than the accuracy in [8], in which only the GPS sensor and GIS data was utilized. In [1], the maximum detection accuracy achieved was 93.8% for the Random Forest (**RF**) model. Consequently, we conclude that with accelerometer sensor data considered in conjunction with GPS/GIS, we saw an increase of over 5% in the detection accuracy of outdoor transportation modes. Thus, using GPS, GIS, and accelerometer is more effective than using GPS and GIS data only. It was also clear that the Random Forest model is slightly more effective than the Decision Trees (**DT**) model; this finding is in-line with the results of [8].

Table 8- F Score of outdoor transportation mode detection (GPS/GIS and accelerometer)

	F-Score					
	T	B	S	W	C	Average
DT	93.5	94.1	100	98.9	96.2	96.4
Bagged DT	94.3	95.7	100	98.9	99.0	97.4
Boosted DT	95.2	97.1	100	98.9	98.1	97.7
RF	99.2	100	100	98.9	99.1	99.4
Bagged RF	99.2	100	100	99.4	100	99.7
Boosted RF	98.4	100	100	99.4	99.0	99.4
Stacked	99.2	100	100	99.4	100	99.7
Vote	99.2	100	100	99.4	100	99.7

Another subtle observation is the fact that for Decision Trees, bagging and boosting techniques improve the detection accuracy by over 1%. In the case of Random Forest, bagging and boosting merely improves the detection accuracy. Also, Staking and Voting meta-level classifiers have the same detection accuracy.

Both stationary and walking modes have very high detection accuracy. This makes sense because using signals from both the accelerometer and GPS sensors clearly distinguishes these modes from all other outdoor transportation modes. For example, for stationary transportation mode, the speed and acceleration computed from GPS should be close to zero and the accelerometer reading in x, y, and z axis should not have high deviation. In the case of walking transportation mode, the speed and acceleration from GPS is low, additionally, the accelerometer high/low peaks and deviation that occurs for each step taken makes these mobility patterns distinct.

Again, we were interested in identifying the most important mobile phone sensor (i.e. GPS or accelerometer) for outdoor transportation mode detection. To quantify which sensor is most effective for outdoor transportation mode detection, we studied the detection accuracy when only GPS/GIS is used compared to when the accelerometer is considered. For both cases (i.e. GPS/GIS only and accelerometer only), the outdoor transportation mode detection accuracy is over 90%. Thus, even though the accelerometer is far more effective than the GPS/GIS model for indoor transportation mode detection, for outdoor transportation mode detection this is not the case. Instead, both models produce almost the same detection accuracy.

5.8.3 Indoor and outdoor transportation mode detection

Outdoor transportation modes (i.e. train (**T**), bus (**B**), stationary (**S**), walk (**W**), and car (**C**)) and indoor transportation mode (i.e. stairs (**ST**), elevators (**EL**), and escalators (**ES**)) detection accuracy is studied in this section. We will present the F Score for each transportation mode. Detecting and distinguishing clearly between outdoor and indoor transportation modes is important for accurate calorie consumption computation. For example, walking, stationary, and climbing stairs consume different amounts of calories.

Based on Table 9, the detection accuracy of all the classification schemes is over 96% on average. This implies that the selected classification feature set using GPS, GIS, and accelerometer is robust. More specifically, the Random Forest (**RF**) model is more effective than the Decision Tree (**DT**) model. Additionally, it is observed that the most accurately detected modes are stationary and elevator. This makes sense since elevators have profound vertical movements, and stationary has close to zero speed and little deviation in the accelerometer signals. Voting achieves close to perfect transportation mode detection for both indoor and outdoor transportation modes. Voting combines the prediction of Decision Trees (**DT**), Random

Forest (**RF**), Naïve Bayes, and Multi-layer Perceptron, by taking the average of the probability distribution of these base-level classifiers.

Table 9-F Score of indoor and outdoor transportation mode detection

	F-Score								
	T	B	S	W	C	ST	EL	ES	Average
DT	90.5	89.0	100	98.6	94.2	100	100	99.2	96.4
Bagged DT	95.2	92.1	100	98.9	94.2	100	100	99.2	97.4
Boosted DT	93.8	97.0	100	97.8	93.2	100	100	99.2	97.7
RF	97.6	98.6	100	98.9	97.0	98.6	100	98.4	98.6
Bagged RF	97.6	100	100	99.4	98.1	98.6	100	98.4	99
Boosted RF	99.2	97.8	98.5	98.9	100	98.6	100	97.5	98.8
Stacked	100	97.2	98.5	98.3	99.0	100	100	99.2	99
Vote	100	100	100	100	100	99.3	100	99.2	99.8

In Table 10, the confusion matrix for the Decision Tree classification scheme is presented. Though Voting was the most accurate model, we present the confusion matrix on the Decision Tree because we wanted readers to observe and understand the cases where misclassifications between transportation modes take place. From the matrix it can be seen that the model can effectively distinguish between indoor and outdoor transportation modes.

Table 10-Confusion matrix of Decision Tree (DT)

	Train	Bus	Stationary	Walk	Car	Stairs	Elevator	Escalator
Train	57	5	0	0	0	0	0	0
Bus	6	61	0	0	2	0	0	0
Stationary	0	0	33	0	0	0	0	0
Walk	0	0	0	90	1	0	0	0
Car	0	2	0	1	49	0	0	0
Stairs	0	0	0	0	0	71	0	0
Elevator	0	0	0	0	0	0	60	0
Escalator	1	0	0	0	0	0	0	60

For outdoor transportation modes, misclassifications in detection occur between buses and trains. This makes sense since buses may drive close to rail line geometries and confuse the detection algorithm. There are also misclassifications for cars and buses. This also makes sense since a car may drive close to a bus (e.g. behind the bus), and hence the location of the car continuously corresponds to the bus's real time location. Consequently, spatial classification features used for supervised learning such as average bus closeness and candidate bus closeness are low.

5.8.4 Mobile phones with GPS or accelerometer turned off

The GPS sensor is more common and accessible on today's mobile phone than the accelerometer. In this subsection, we study the accuracy of transportation mode detection with the assumption that either the accelerometer or the GPS sensor is turned off. This will occur, for example, in case that a traveler turns off their GPS sensor to conserve mobile phone battery power or in the case where GPS fails such as in canyons or under bridges.

From Table 11 it can be seen that in the case where the GPS sensor on the mobile phone is turned off and only the accelerometer sensor is utilized, the average transportation mode detection accuracy is still over 90%. On the other hand, when suppressing accelerometer sensor reports and using only GPS/GIS, the detection accuracy reduce to below 83%.

Consequently, with accelerometer sensor reports only, the system is still efficient and performs better than solely GPS. On the other hand, the best performance is obtained when readings from the two sensors are combined with GIS information.

This finding of high detection accuracy of outdoor transportation modes, particularly among motorized transportation modes, using only the accelerometer sensor was unexpected, but the results conform to previous studies.

Table 11-Average precision when sensors are turned off for indoor and outdoor modes

	Average precision for all transportation modes	
	GPS turned off	Accelerometer turned off
DT	91.4	78.3
Bagged DT	92	79.8
Boosted DT	92.5	79.3
RF	92	82.4
Bagged RF	92.2	81.6
Boosted RF	92.4	82.4
Stacked	92.5	82.6
Vote	92.7	80.4

For example, the work in [43] utilizes only the accelerometer on the mobile phone and distinguishes between buses and trains with over 90% accuracy. Likewise, even though [2] did not consider indoor modes, they distinguish between motorized modes (i.e. buses, cars, and trains) with over 83% accuracy on average using only accelerometer. Intuitively, trains are moving at relatively stable speed with few abrupt sharp turns. In contrast, the buses are typically moving with sharp turns and frequent acceleration and deceleration as observed by the authors of [43]. These differences in mobility patterns can be captured by the accelerometer sensor and then utilized to distinguish transportation modes.

5.8.5 Battery power consumption

Utilizing the sensors on a mobile phone consumes battery power. In this section, we study the battery duration in relation to the utilization of the GPS and accelerometer sensors. For battery power consumption, we studied the battery exhaustion of an Apple IPhone 3G while running the system under two cases:

Case 1 - Both GPS and accelerometer sensor reports are submitted over the Internet every two seconds for classification feature creation. The GPS sensor is accessed once every two seconds and the accelerometer is continuously accessed.

Case 2- Only accelerometer sensor reports are submitted over Internet every two seconds for classification feature creation. The accelerometer sensor is continuously accessed.

The setup is as follows. First, we make sure that the application is the only external application running on the IPhone 3G and that the device is unlocked for the duration of the experiments. We then charge the IPhone until the battery is full to capacity for both cases. Next, we start the application under case 1 followed by case 2. The time it takes to fully exhaust the battery is then observed for both cases. Based on the battery exhaustion time and the start time, a value for battery exhaustion is computed as the time difference.

While there are standard equipment such as hardware power meters (e.g. Monsoon Power Monitor [84]) that measures the power consumption of an application running on a mobile device, our intention was to measure how long it takes the transportation mode detection software to completely exhaust the battery under varying sensor fusion combination.

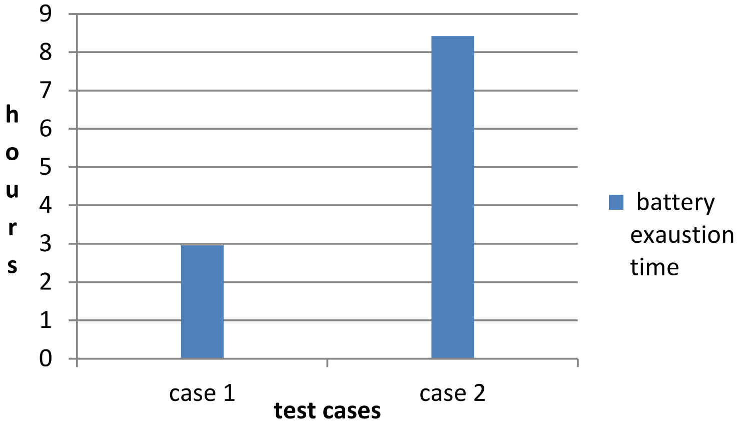


Figure 27- Battery power consumption and sensor usage (Apple's IPhone 3G)

From Figure 27, when only accelerometer reports are submitted by the traveler, the battery life is longer. In this scenario where only accelerometer reports are submitted, the transportation mode detection accuracy is over 92% accurate. This high detection accuracy while using only the accelerometer data is due to the fact that the accelerometer classification features utilized in this work are more effective than GPS/GIS data.

Thus, the traveler could turn off their GPS sensor to save battery power, and still achieve high transportation mode detection accuracy. This property is desirable, since the traveler's location is not submitted to the central server, then the system does not suffer from possible location privacy breaches.

5.9 Limitations of study

In this study, there are several limitations. One limitation of this approach is the constraint on mobile phone position. In the future, we will study how different mobile phone orientations, placements, and noisy sensor data affect the transportation mode classification accuracy.

Another limitation is the size of the dataset considered. Realistically, we need to collect more sensor data (GPS and accelerometer) in an uncontrolled manner from several travelers to effectively study and validate the proposed model. The dataset considered provided an insight on indoor and outdoor transportation mode activity classification, however with more sensor data including noise, the classification accuracy may be reduced. For larger scale experiment, one may consider switching to Android phone which allows background applications to collect sensor data. Such that the subjects can still use their phones as it is and generate more realistic data in the wild. Leaving the phone unlocked with data collection application running on the foreground is for a proof of concept, but not the best option for a longer term study.

5.10 Discussion

In the proposed work, we utilize two sensors on the traveler's mobile phone along with transportation network geospatial data to effectively detect both indoor and outdoor transportation modes. Combining these two sensors with geospatial knowledge increases the outdoor detection accuracy by over 5.5% compared to using only GPS and GIS as done in our prior work [8]. The use of several meta-level classifiers such as stacking and voting formed by combining several base-level classifiers are found to be slightly more effective than the traditional base-level classifiers by themselves such as Decision Trees and Random Forest

Further evaluation reveals that the accelerometer sensor is more effective than GPS for indoor transportation mode detection. Using the accelerometer only, the system can still achieve over 95% detection accuracy. Using only the GPS sensor and geospatial knowledge of the transportation network such as real time bus location, the indoor transportation mode detection accuracy reduces to below 67% on average.

For performance purposes, several spatial indexing strategies on the geospatial data used in this work are evaluated. Results indicate that a combination of R-Trees and GIST spatial indexing is the most effective spatial indexing technique.

In terms of battery power consumption and sensor reports, experimental results on an IPhone 3G indicate that using both the GPS and accelerometer sensors exhaust the battery power much faster than using only accelerometer sensor reports. Thus, one could turn their GPS sensor off, conserve battery power, and still achieve a high level of accuracy for transportation mode detection service. Using only accelerometer reports and pruning GPS reports will reduce the mobile phone's battery exhaustion rate by nearly 3 times.

In general, we combine multiple common mobile phone sensors to derive new classification features that are effective for both indoor and outdoor transportation mode detection. This strategy of using GPS sensor, accelerometer sensor, real time bus locations, rail line location, and bus stop locations is new, highly effective, and robust under every day usage. The proposed scheme can achieve 98% detection accuracy across the modes cars, buses, trains, walking, stationary, stairs, elevator, and escalator. This 98% represents a 5.5% increase in detection accuracy when compared to our prior model that considers only outdoor modes while utilizing the GPS sensor with GIS spatial and temporal properties of the underlying transportation network. The combination of GPS, GIS, and accelerometer is promising, it can distinguish between indoor, outdoor, and various different categories of motorize transportation modes. This is not the case for previous work on activity recognition.

Furthermore, from experiments for indoor transportation mode detection, the accelerometer only sensor model is far more effective than the GPS/GIS model. For outdoor transportation mode detection, the difference in detection accuracy between both models (i.e. GPS/GIS and accelerometer) is not as distinct as in the case of indoor transportation mode. In general, this work shows that the most effective model for indoor and outdoor transportation mode detection on mobile phones is the combination of accelerometer, GPS, and GIS.

In the future, we will study how different mobile phone orientations and placements, affect the transportation mode classification accuracy. In addition to mobile phone orientation studies, the conservation of battery power by selectively sampling the mobile phone sensors or adaptively controlling the sampling frequency is also a subject of future work. For example, since GPS does not contribute to indoor transportation mode detection accuracy, then when the traveler is indoors we can deactivate GPS. Another area to explore in the future is the use of other sensors such as camera, NFC, and Bluetooth for transportation mode activity detection from smartphones.

5.11 Relevant work

In [12, 13], the authors proposed an unsupervised learning technique to detect the traveler's transportation mode. The modes considered in [12, 13] include bus, car, and walk. In addition to identifying the transportation modes used, [12, 13] can identify the traveler's goals, such as trip purpose and trip destination. The GPS sensor along with the travel history of the traveler such as where they parked their car or board a bus is used in [12, 13]. The proposed work is different since the travel history of a person such as where they parked their car is not considered.

Additionally, we use a supervised learning model and consider more sensors (i.e. accelerometer). Also, the present work can detect both indoor and outdoor transportation activities.

The work of Zheng et al [14, 15] used only the GPS sensor to infer transportation mode. They proposed a post processing strategy and a robust set of classification features that make the model rigid to traffic and weather changes. The present work is different because we consider more sensors and transportation network information. Also, we focus on both indoor and outdoor transportation modes, while [14, 15] focuses on a subset of outdoor transportation modes.

The Jigsaw sensing system proposed in [41] balances the performance needs of the application and resource demands of continuous sensing on mobile phones. Our work is different since [41] focuses solely on outdoor transportation mode and does not consider GIS information such as real time bus locations and bus stop spatial properties. Also, [41] does not distinguish between motorized transportation modes such as buses and trains. Another energy aware mobile sensing transportation activity recognition system is [42]. The authors of [42] combined the minimal set of sensors such as GPS, accelerometer, microphone, or Wi-Fi. The modes that [42] detects are still, walking, and vehicle. I.e. [42] does not distinguish between types of vehicles nor does it detects indoor modes such as elevator or escalator.

The PEIR system [1] considered a Hidden Markov Model (HMM) for transportation mode detection and utilizes GPS, GSM, and open source weather and traffic information. In addition to transportation mode detection, the work in [1] also measures environmental impacts (e.g. carbon impact) and exposures (e.g. smog). This dissertation is different from [1], in that we considered the accelerometer sensor and a wider variety of transportation modes.

The work in [43] focuses on predicting bus arrival times from crowd-sourced information obtained from the mobile phones of bus riders. In [43], the authors utilize the accelerometer sensor on travelers’ mobile phones to distinguish between buses and trains, thus determining the source of the crowd-sourced information. In our work, we used more sensors, GIS data about the transportation network, and we detect both indoor and outdoor modes. In addition, we distinguish among more categories of motorized transportation modes.

Alternatively [2] estimates real time CO_2 emissions using inertial information obtained from the mobile phone’s accelerometer sensor. In order to quantify the amount of CO_2 emissions, the authors computed a function of transportation mode and distance. The transportation modes detected in [2] are outdoor modes. Our work is different, we utilize the GPS sensor, GIS, and also considered indoor transportation modes such as elevators and escalators. The state of the art that used both GPS and accelerometer sensors on the mobile phone for transportation mode detection is [11]. However, [11] does not distinguish between motorized transportation modes such as car, bus, and train. Distinguishing between motorized modes is a realistic and important problem for navigation and real time traffic providers such as Google or Nokia Location and Commerce (formerly NavTeq). These companies collect data from probes, data coming from travelers on buses, cars, and trains are handled differently for real time traffic analysis. Thus, while [11] only detects still, walk, run, bike, and general motorized modes. This work distinguishes between motorized modes such as car, bus, and train, and also considers indoor modes.

The state of the art that combines GPS and GIS data for transportation mode detection on mobile phones is [28]. Additionally, in [28], if a traveler is traveling by public bus, the system can provide further information on which particular bus she is riding. The drawback of [28] is that it does not consider the accelerometer sensor and it does not detect indoor transportation modes such as elevator and stairs.

6. PARKING STATUS DETECTION (PSD)

Finding street parking in crowded urban areas is a challenging chore that most drivers dislike. In one business district of Los Angeles, researchers found that vehicles searching for parking created an equivalence of 38 trips around world, produced 730 tons of carbon dioxide, and burned 47,000 gallons of gasoline [19, 51] in one year. Real-time parking availability information has been shown helpful in reducing parking discovery time (see e.g., [51]). Presently real-time parking availability information is usually collected from fixed sensors. The SFPark project in San Francisco [17] implanted sensors under road surfaces. These sensors cover 6,000 parking spaces which is about 25 percent of available street parking spaces. The total project volume together with smart parking management functions is 23 million dollars [17, 19] and each sensor cost approximately US $500. Another project [19] uses ultrasonic sensors that are externally mounted on car side-doors to detect parking spaces. The cost of the system for each car is approximately US $400. Additionally, the manual reporting of empty street parking spaces when one sees them is cumbersome for drivers.

In this dissertation, we propose PSD a solution called PhonePark to detect parking/deparking activities using sensors in mobile phones, such as GPS, accelerometer, and Bluetooth. External sensors such as those implanted under road surfaces [17], installed in parking lots, or attached to side-doors of vehicles [19] are not needed in our approach.

6.2 Parking status detection

We propose three methods for a PSD to detect parking status (1) Mobile phone's Bluetooth connection to in-vehicle Bluetooth, (2) transportation monitoring, and (3) parking pay by phone piggyback.

6.2.1 ***Connection to in-vehicle Bluetooth***

This method utilizes the Bluetooth connection between the driver’s mobile phone and the car to detect parking/deparking activities. Nowadays more and more cars have in-vehicle Bluetooth capability, which allows the owner of a car to register her mobile phone to the in-vehicle Bluetooth system. Once the mobile phone is registered, whenever the owner is inside the car and the engine is started, the mobile phone is automatically connected to the in-vehicle Bluetooth system. On the other hand, when the engine is stopped or the owner goes away from the car for a short distance (□10 meters), the Bluetooth connection between the mobile phone and the car is broken. Thus, if the Bluetooth connection between the driver’s phone and the car is broken, then it can be inferred that the driver parks the car. If the Bluetooth connection is established, then it can be inferred that the driver deparks. This concept is illustrated by Figure 28. The Bluetooth ID of the car can be used by the PSD to distinguish between the Bluetooth connection with car and that with other devices such as a Bluetooth headset.

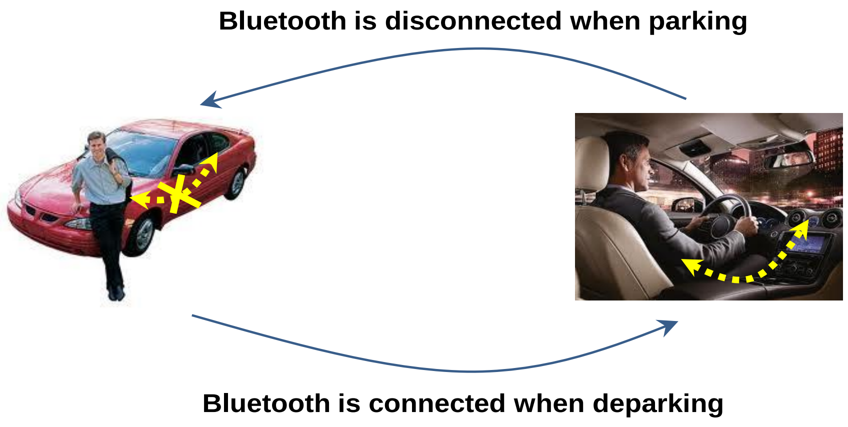

Bluetooth is disconnected when parking
Bluetooth is connected when deparking

Figure 28- Parking detection using Bluetooth connection between phone and car

6.2.2 Transportation mode monitoring

This method employs a transportation mode detection algorithm to track the transportation mode of the driver. It monitors the transitions of transportation mode and infers parking status from a sequence of transitions. For example, if the transition sequence driving → stationary → walking is observed, then the PSD infers that the driver parked at the stationary point. Similarly, if the transition sequence walking → stationary → driving is observed, then the PSD infers that the driver deparked at the stationary point. This concept is illustrated by Figures 29 and 30.

In the case that the output of the transportation mode detection system is from the set {car, walk, stationary} then the output is fed into a parking detection middleware. The objective of the parking detection middleware is to observe when and where a traveler parks or deparks her car.

Let a traveler u use a particular transportation mode $tmode$ at time t, and denote this traveler byu_t^{tmode}. The parking detection middleware is initiated if $tmode$=car. Then the middleware tracks the following transportation mode transitions, $u_{t0}^{car} \rightarrow u_{t1}^{stationary} \rightarrow u_{t2}^{walk}$ such that $t_0 < t_1 < t_2$. When this mobility transition flow is observed, the middleware checks whether the traveler is a driver or passenger. If the traveler is a driver, then it is concluded that the traveler parked. More specifically, the traveler parked at the stationary point at time t_1.

The parking detection middleware can be represented by the deterministic finite state machine M in Figure 29. M consists of four tuples: Q, Σ, δ, q_0. Q is a finite set of states that are related to parking (i.e. driving, stop, off-car, park, unpark, and passenger).

The finite sets of input Σ to M are car, walking, and still/stationary transportation modes. The transition function of M is $\delta: Q \times \Sigma \rightarrow Q$. The start state q_0 is the driving state.

The state machine works as follows. When the car transportation mode is detected, the middleware is activated and is aware that the traveler is driving. With subsequent car mode inputs, the traveler remains in the driving state. With a stationary mode input, the traveler enters the stop state. This transition from driving state to stop state does not necessarily imply parking. For example, in traffic or at a stop light, a traveler may transition from car mode to stationary mode (i.e. driving state to stop state). Observe that if the traveler is in the driving state, a walk, bus, or train mode input is not possible since she must stop the car before starting to walk or board a bus/train in real life.

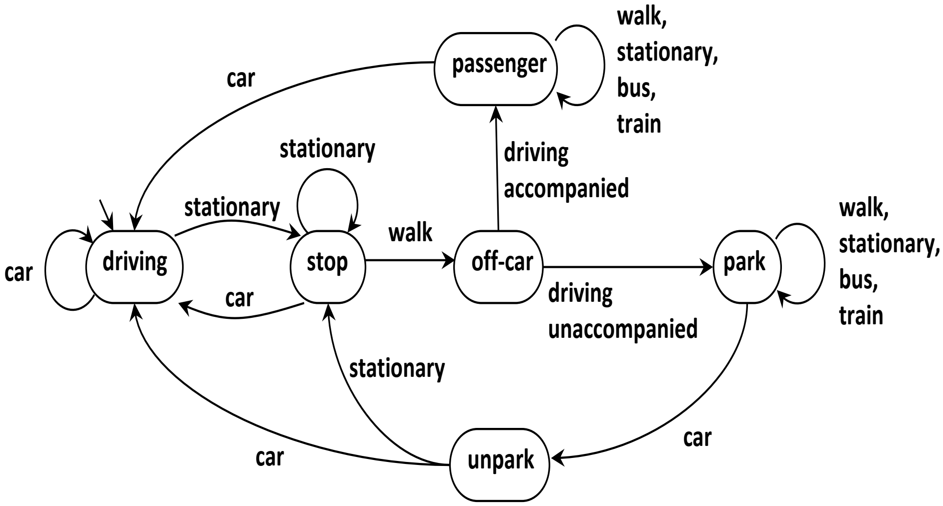


Figure 29-Parking finite state machine

When in the stop state, with subsequent stationary mode inputs from the transportation mode detection system, the middleware remains in the stop state. However, if in the stop state and receives car mode inputs, it transitions back to the driving state. Observe that we cannot go directly from the driving state to the park state. For parking, the driver is expected to alight from the car after stopping.

Finally, if in the stop state walk inputs are received, it is an indication that the traveler may have parked. However, before making the conclusion, the machine enters an off-car state. In the off-car state the system determines whether the traveler is a driver or a passenger of a car. There are several algorithms that distinguishes between drivers and passengers of a given car [85]. If the traveler is determined to be a passenger, then the machine enters the passenger state. The machine remains in the passenger state until a car mode input is received. Given the car mode input, the machine transitions back to the driver state.

If in the off-car state the traveler is determined to be a driver, then the traveler has parked at the spatial and temporal properties of the previous stop state. In this case, the system generates a report indicating that the parking space is occupied. The generated report can be plotted on a map of parking availability.

Once in the park state, with a subsequent walk, stationary, bus, or train mode input, the finite state machine remains in the park state. However, if the finite state machine (i.e. M) is in the park state and receives car mode inputs, then the finite state machine transitions to the unpark state. When this occurs, it implies that the driver has deparked her car. In this case, PhonePark generates another report indicating that the corresponding parking space is now available (See Figure 3). This technique enables one to compute how many cars are parked on a block if every traveler is using the system.

The reliability of this method is enhanced by detecting the activity of going to a nearby pay-box (as in Chicago Parking see [86]), paying, and returning to the car to place the ticket on the dashboard (see our prior work in this direction in [28]). Using only transportation mode transitions, we achieved over 85% accuracy for parking/deparking detection on the PSD. Combining this with Bluetooth and pay-by-phone piggyback can improve the accuracy further.

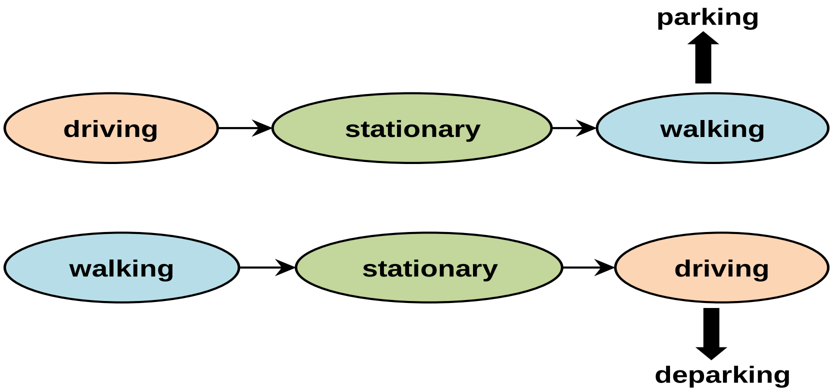


Figure 30-Simplified parking/deparking detection based on transportation mode transitions

6.2.3 Pay-by-phone Piggyback

This method takes the advantage of the pay-by-phone parking payment service. This service allows a driver to pay for parking by entering to her mobile phone the parking slot number (posted on parking meters) and parking duration [20, 21]. The information is then submitted from the mobile phone to a payment service system. The pay-by-phone may help parking detection in the following way. With the mobile phone user’s agreement, the PSD (which resides on the same phone as the pay-by-phone application) monitors the user’s usage of pay-by-phone service. When the user submits the payment, the PSD infers that the user has parked (see Figure 31). The PSD may possibly overhear the parking duration input and thus predict the time of deparking. The pay-by-phone service has been available in over 30 cities in US including San Francisco, Chicago, Washington DC, etc. (see [20, 21]).

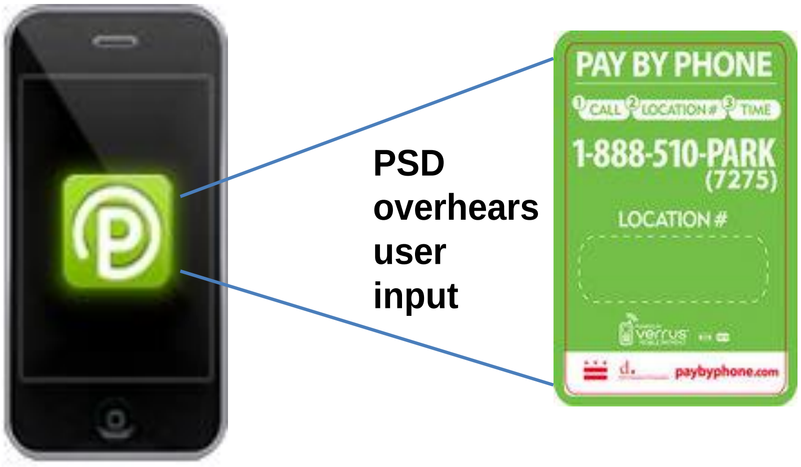


Figure 31-Parking status detection by pay by phone piggyback

6.3 Energy conservation study

When PhonePark is running continuously (i.e. both indoors and outdoors) on an IPhone 3G, the battery exhaustion time varies with the window size (i.e. sensor sampling period). With a window size of 2 seconds, the battery exhaustion time is over 3 hours. However, with a window size of 7 seconds, the battery exhaustion time is over 4.5 hours. Thus, in PhonePark it makes sense to adaptively control the window size. Moreover, the energy can be conserved by activating PhonePark only when a user is in an outdoor environment. One way to distinguish indoor from outdoor is to sample the GPS and then observe the quality of the location information. For the Bluetooth parking detection, the system consumes less battery power. Bluetooth uses less energy than GPS, hence we can activate or deactivate the system only when the mobile device’s Bluetooth sensor is paired or unpaired with the car’s Bluetooth component.

6.4 Parking and deparking detection accuracy

In this section, we present real world evaluation of the accuracy of the proposed parking status detection system. For these experiments, only parking status detection via transportation mode transition is evaluated.

6.4.1 Automated parking detection accuracy

For the evaluation, multiple drivers using PhonePark drove around Chicago, parked, and then observed if the system determined that they had parked. From the experiments, the matrix recall parking accuracy is computed as: *parking accuracy = number of detected parked instances by PhonePark / number of actual parked instances.*

In Figure 31, 5 experiments on participants' parking detection accuracy are presented. From Figure 31, it can be seen that the proposed parking detection mechanism detects parking with over 80% accuracy on average.

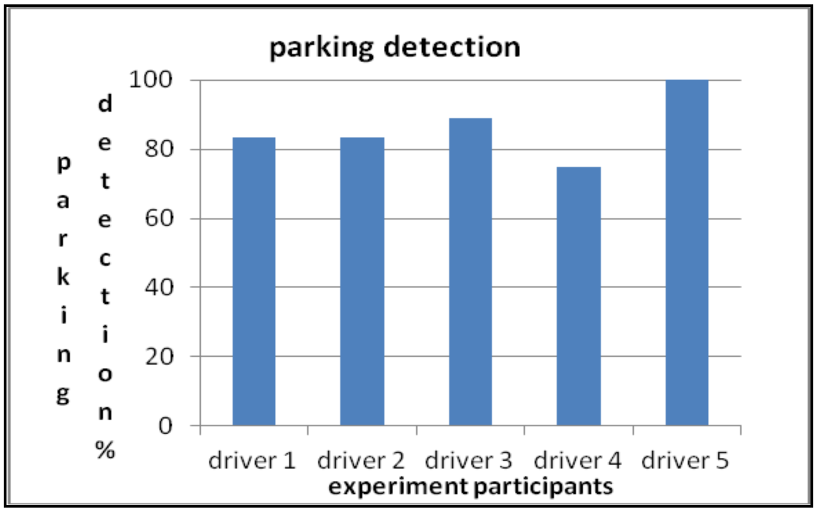


Figure 32- Parking detection accuracy

In some cases PhonePark can achieve high parking detection accuracy. For example, driver 5 achieved high accuracy. In rare cases, the traveler alights from their car very quickly (i.e. less than the time window) and PhonePark did not detect the stationary transportation mode (i.e. stop state). Thus, the parking middleware did not confirm parking since the stop state was expected before the parking state. Also, if one drives a short distance quickly, then parks, if PhonePark does not detect the driving state, parking is not detected.

For driver 4 in Figure 31, the experiments were conducted in the Downtown region where the skyscraper concentration is highest in Chicago. In this region, the GPS inaccuracy is also very high and may mislead the system. For example, in some cases when driver 4 parked, because of the GPS inaccuracy in this region, the middleware remains in the driving state. In this case the middleware was over 75% accurate for automated parking detection. However, this region is a small percentage of the entire city.

In general, the proposed hierarchical parking strategy is effective for parking detection. It can achieve over 85% accuracy for automated parking detection.

6.4.2 Unparking or deparking detection accuracy

Whenever a driver parks a car, PhonePark generates a report indicating that a street parking space is occupied. Likewise, whenever a driver deparks a car, PhonePark generates another report indicating that a street parking slot is available. The addresses and timestamps of these reports are cached. In this section, PhonePark's ability to detect that a driver deparked a car is evaluated. For unparking, the driver transitions from parking state to driving state (i.e. park → driving) or enters the driving state initially. From the experiments, we compute the matrix recall unparking accuracy as: *unparking accuracy = number of detected depark instances by PhonePark / number of actual unparks instances.*

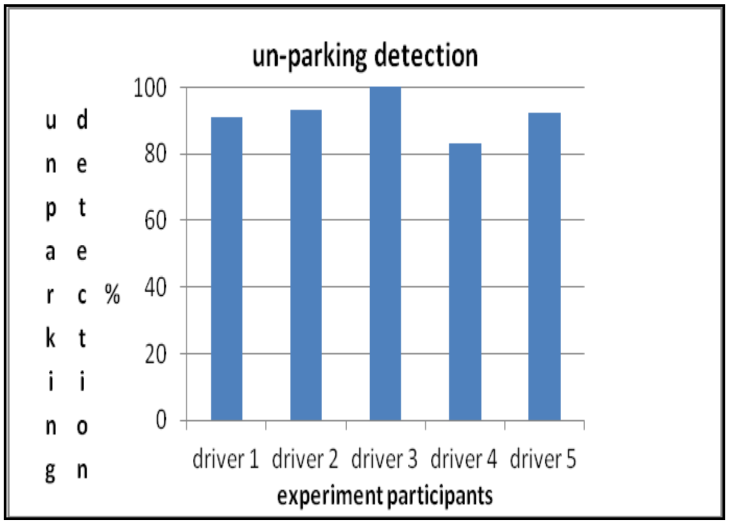
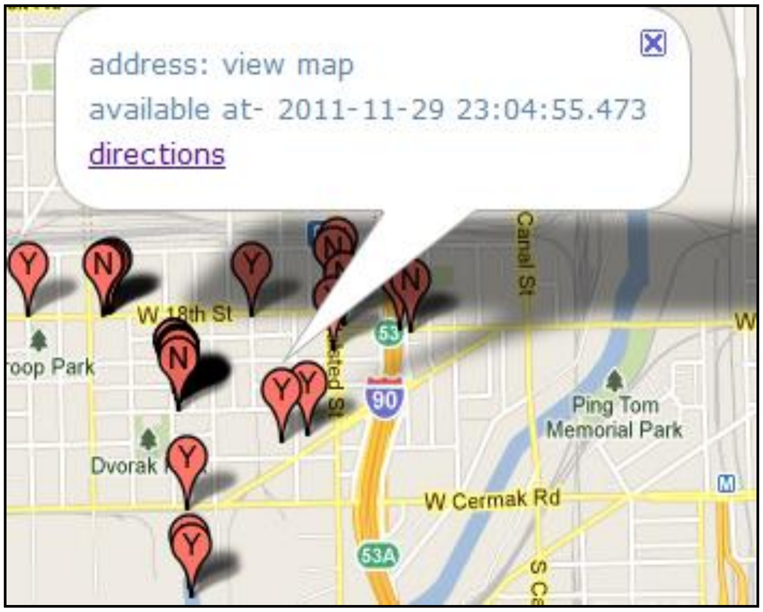


Figure 33-unparking or deparking detection accuracy

In Figure 32, the unparking detection accuracy for the experiment participants is presented. In general, the unparking detection accuracy (i.e. over 90% on average) is higher than the parking detection accuracy. For unparking, the traveler enters the car state for the first time, or from the park state. Unparking is easier to detect than parking.

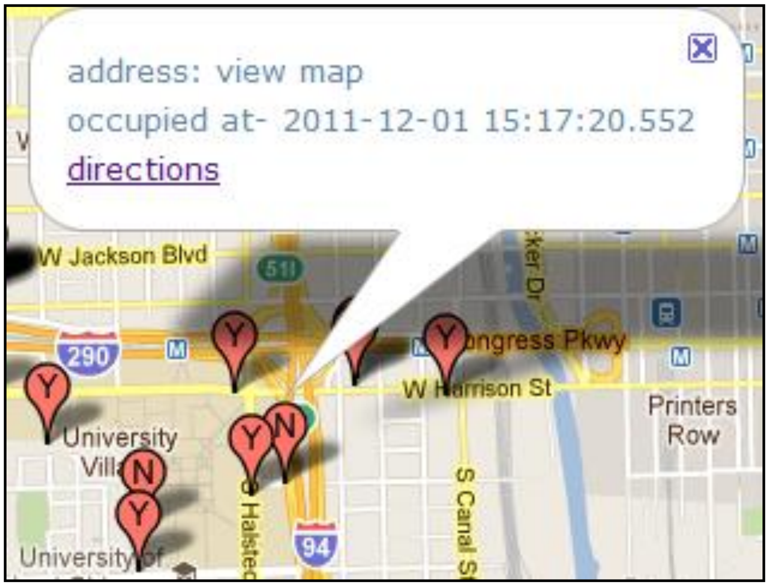


address: view map

available at- 2011-11-29 23:04:55.473

directions

Figure 34- PSD generates spatial and temporal parking and deparking reports



address: view map

occupied at- 2011-12-01 15:17:20.552

directions

Figure 35-PSD generates spatial and temporal parking and deparking reports

In Figures 33 and 34, based on the parking and unparking detection reports that are generated by the PhonePark parking status detection system, PhonePark creates a map of availability of street parking spaces. For example, in Figure 33, a street parking space is available at 11:04 pm on the 29th of November 2011. Likewise, in Figure 34, a street parking space was occupied on December 1st 2011 at 3:17 pm.

6.5 Relevant work

Parking garages uses in-out counters at exit and entry points to compute the number of additional vehicles they can accommodate [24]. There exist other systems that allow travelers to access parking information and make prior reservations for parking in areas such as airports and rail stations [46, 47]. PhonePark is different and focuses on street parking.

The GPS sensor has been used in the past for parking guidance. For example, GPS guided parking is used in [48]. Our work is different, using GPS, we detect that a traveler parks or deparks automatically.

The availability of the vacant parking spaces can be calculated by external sensors such as those installed in the parking areas, which counts the number of cars which enter and exit from the parking space [17]. For example, in San Francisco, several parking projects have been initiated [17] [20] that utilizes externally implanted sensors. However, this strategy is expensive to deploy and maintain. For example, in one project that covers 6000 parking spaces the cost was over 3 million dollars [17, 19, 51].

Ultrasonic sensors at the top of each parking space or on vehicle side-doors [19] can be used to sense the availability or unavailability of each parking space. These sensors update the central database. Our work is different; we ignore external sensors and consider only sensors available on the travelers' mobile phones to concur when and where a traveler had parked their car automatically. Sensors implanted under road surfaces or attached to the car side-doors are expensive to deploy and maintain (e.g. [17] cost US $500.00 per system for each parking space, and [19] cost US $400.00 per system for each car). These sensors may underperform in extreme weather. For example, in heavy snow these sensors may be covered. Using mobile phones is cheaper, more convenient, and more flexible.

Google's approach to parking status detection is called OpenSpot [56, 57] and relies on drivers to manually report an empty parking spot when they depark or see one available. This system is cumbersome and drivers don't feel comfortable to use their mobile phones for manual reporting while driving. The proposed approach in this thesis automates parking status detection and manual entry of available street parking spaces is not required.

Yan et al. [86] try addressing this parking status detection problem by designing an online auctioning system around the availability of a parking spot, offering financial incentives to participating users. The drawback of this approach is that it still requires manual user intervention for the system to be useful.

Nawaz et al [87] considers a smartphone sensing system that utilizes Wi-Fi beacons to detect parking activities. Using Wi-Fi beacons is interested since Wi-Fi uses less energy than GPS. The work in [87] is different from this thesis since in the proposed approach, we did not consider Wi-Fi.

Coric et al [88] focuses on deriving parking maps from historic sensor data did not detect parking activities in real-time. Hence [88] is out of scope from this parking status detection research.

7. PARKING AVAILABILITY ESTIMATION (PAE)

Now consider how PhonePark would use observations of mobile phones to estimate the number of available parking spaces in a street block. There are two challenges for this problem. First, PhonePark may produce false observations of parking activities due to GPS errors, transportation mode detection errors, and Bluetooth pairing errors. Second, not all drivers are equipped with PhonePark and therefore the observations of PhonePark represent only a portion of the full picture. In other words, the penetration ratio of the PhonePark enabled mobile phones may be low, so an estimate is needed for the determining parking availability on a street block.

In this dissertation, we introduce an algorithm to construct the *historical availability profile* (HAP) for a street block. This algorithm computes estimated values for the mean and the variance of parking availability for an arbitrary street block. In addition to the HAP construction algorithm, we also introduce other algorithms that can estimate the real-time parking availability on a street block. These real-time estimation algorithms are called *parking availability estimation* (PAE) algorithms and they utilize different combinations of the historical parking availability and real-time observations to estimate the current street parking availability.

In summary, this chapter makes the following scientific contributions: (1) Algorithms to detect the status of street parking slots. These include parking/deparking detection by transportation mode transitions, Bluetooth pairing, and parking pay-by-phone piggyback, (2) An algorithm that can construct the historical availability profile (i.e., mean and variance of parking) for a given street block, (3) Algorithms for estimating in real-time the current parking availability on a street block, (4) Experimental evaluation of the algorithms using actual real-time street parking data from SFPark.org, and (5) Theoretical underpinnings of the proposed techniques.

7.2 Overview of parking availability estimation

The estimation software system consists of two components, namely *parking status detectors* (PSDs) and the *parking availability estimator* (PAE) . A PSD runs on a mobile device. It detects when and where a driver parks/unparks and whether she pays for parking. When the PSD recognizes a parking or deparking activity, it submits a report to the PAE component indicating that a parking space is occupied or released[2]. The PAE component aggregates the reports of individual PSDs to estimate the number of available parking slots in each street block. For estimation, PAE considers the facts that not all drivers are equipped with the PSD component (i.e., the market penetration ratio of PhonePark is not 100%) and the mobile device can have false positive and false negative possibilities for parking detection. Furthermore, to compensate for the inaccuracy of PSD observations, it combines PSD observations with historical parking statistics to get the final estimate. Historical parking statistics is obtained from the proposed HAP construction algorithm while PSD observations are obtained from the drivers with the PSD devices. See Figure 35 for more details.

---

[2] The submission of a parking/deparking activity can be anonymized and incentivized. In general, privacy and incentives are orthogonal issues that are not addressed in this paper.

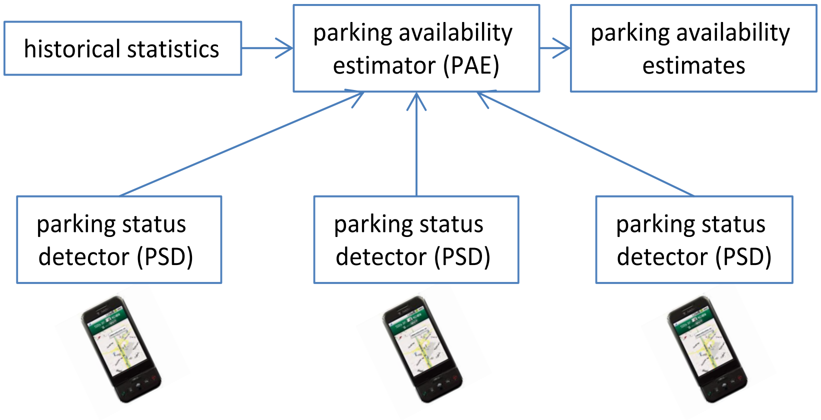

historical statistics
parking availability estimator (PAE)
parking availability estimates
parking status detector (PSD)
parking status detector (PSD)
parking status detector (PSD)

Figure 36-Components of the PhonePark estimation system

An observation of the PSD may be incorrect due to many reasons. First of all, the classification of the transportation mode may be wrong due to sensor errors and ambiguity between different transportation modes. For example, the mobility pattern of walking is similar to that of driving in a heavy traffic situation. Second, even the transportation modes are correctly tracked, the inference from the transition sequence may be wrong. For example, the transition sequence *car* → *stationary* → *walking* may be the result of a passenger being dropped off (as opposed to a driver parking). Likewise, Bluetooth pairing or pay-by-phone piggyback can have errors.

All the errors of PSD observations propagate to the PAE component when PAE aggregates these observations to estimate the number of available parking spaces. Furthermore, when PAE scales PSD observations to compensate for the market penetration ratio, there are sampling errors. PAE takes all these errors into account when combining the real-time observations with historical statistics. Recall, historic statistics are computed by the *historical availability profile* (HAP) algorithm.

Apart from studying the effectiveness on real-time parking estimation of using the historical statistics and the scaled real-time observations solely, we also consider the combination of both the historic statistics and the real-time observation. The combination strategies are: (1) Using a fitted weight scheme and (2) Using an adaptive Kalman Filter scheme. These four schemes are compared with the true real-time parking availability in the evaluation section.

For the fitted weight scheme, based on the market penetration ratio of the PhonePark enabled mobile phones, false positive, and false negative probabilities of the PhonePark mobile devices most effective weight can be determined beforehand. Using this predetermine weight, we now have the most effective combination of historic statistics and real-time observations.

For the Kalman Filter scheme, the combination is essentially a weighted average between the parking availability detected by PSDs and the historical statistics as obtained from HAP where the weights are proportional to the accuracies of the two availability sources. The variances represent the two accuracies. In this dissertation we use a Kalman filter, called *adaptive, limited memory filter* (ALMF) [89], to estimate the accuracy of the parking availability detected by PSDs. ALMF estimates the accuracy based on the gap between the historical statistics and the recent PSD observations. It uses this gap to capture the collective effect of all kinds of errors in the PhonePark system. Despite its simplicity, ALMF has been shown effective in situations where error statistics are unknown a priori, changing over time, or hard to analyze (see e.g., [89, 90]). Since errors are handled by PAE, the process of the PSD is deterministic.

The Kalman Filter based weighting between PSD-detected parking availability and historical statistics captures the following intuitions: (i) PSD-detected parking availability is more useful when the market penetration ratio is high and when parking availability is unpredictable from historical statistics; and (ii) historical statistics are more useful when PhonePark is inaccurate.

7.3 Constructing Historical Profiles

The historical profile of a street block contains the mean and variance of the block's parking availability as a function of time. In this section, we discuss how to construct a historical profile from historical PSD observations.

7.3.1 Historical Availability Profile

I. Periodically a street block S experiences a status such that all the parking slots in S are available. For example, there may be a limitation that parking is prohibited from 8am to 9am of each day, or there is street cleaning every Tuesday from 1pm to 3pm during which parking is prohibited. When parking is prohibited, all parking slots are available. The time period during which parking is permitted is referred to as a permitted period. For example, if parking is prohibited from 8am to 9am of each day, the permitted period is from 9am of a day to 8am of the next day.

Time is discrete and proceeds by an atomic unit (e.g., second). Define the *parking availability* of *S* to be the number of available parking slots in *S*. We assume that the time unit is small (e.g., second) such that the parking availability can only change at the beginning or end of a time unit but not during it. A *historical availability profile* (HAP) for *S* includes the average and variance of parking availability of *S* at each time point of the permitted period.

We distinguish between two types of PSD observation errors. One is called *false negative* which occurs when a parking or deparking activity is not detected by a PSD the owner of which performs the activity. Another type is called *false positive* which occurs when a PSD detects an activity that does not actually happen. The probability that a false negative error occurs is called the *false negative probability* and denoted by *fn*. *fn* is a fixed parameter. The probability that a false positive error occurs is called the *false positive probability* and denoted by *fp*. *fp* may vary depending on how the observation is generated. *fp* is attached to the PSD report.

7.3.2 HAP Construction Algorithm

Denote by *N* the total number of parking slots on the street block *S*. Denote by *b* the penetration ratio of PhonePark. The HAP construction algorithm starts with the beginning of a permitted period. At this time point the parking availability is equal to *N*. Then when a parking report is received, the parking availability is decreased by $\frac{1-fp}{b\cdot(1-fn)}$. This is called *scaled decrease*. The intuition behind scaled decrease is as follows. Out of all vehicles that park, only fraction *b* of them are equipped with PhonePark and out of these PhonePark equipped vehicles only fraction $(1-f_n)$ detect the parking activity and send parking reports. Thus, each received parking report represents $\frac{1}{b\cdot(1-fn)}$ actual parking activities. On the other hand, due to false positive errors, only fraction $1-fp$ of received parking reports represent actual parking activities. Similarly, when a deparking report is received, the parking availability is increased by $\frac{1-fp}{b\cdot(1-fn)}$. This is called *scaled increase*. In this way we obtain an estimated function that describes how the parking availability changes over time for the considered permitted period. We repeat this procedure for a number of permitted periods and average among all the obtained functions. Specifically, denote by $\hat{a}_i(t)$ the parking availability estimated by scaled decrease or increase at time *t* for the *i*-th permitted period. Denote by $\hat{q}(t)$ the estimated average and by $\hat{Q}(t)$ the estimated variance of $\hat{a}_i(t)$, respectively.

$$\hat{q}(t) = \frac{\sum_{i=1}^{m} \hat{a}_i(t)}{m} \tag{4.1}$$

$$\hat{Q}(t) = \frac{\sum_{i=1}^{m} (\hat{a}_i(t) - \hat{q}(t))^2}{m} \tag{4.2}$$

where m is the number of permitted periods for which parking availability is collected. The HAP algorithm stops when a certain level of confidence for the accuracy of $\hat{q}(t)$ is reached, e.g., the difference between $\hat{q}(t)$ and the true average parking availability at the time t is smaller than 2 with 95% confidence. In the rest of this section we discuss how the confidence is computed.

7.3.3 Computation of Confidence

Let $\{PP_1, PP_2, \ldots, PP_m\}$ be a sequence of permitted periods for a block S for which parking availability has been collected. We make the following denotations:

$a_i(t)$: the true parking availability at the time t of the PP_i.

$P_i(t)$: the true number of parking activities starting from the beginning of PP_i until time t of PP_i.

$D_i(t)$: the true number of deparking activities starting from the beginning of PP_i until time t of PP_i.

Clearly, $a_i(t) = N - P_i(t) + D_i(t)$.

$p_i(t)$: the number of parking activities detected by PhonePark starting from the beginning of PP$_i$ until time *t* of PP$_i$.

$d_i(t)$: the number of deparking activities detected by PhonePark starting from the beginning of PP$_i$ until time *t* of PP$_i$.

7.3.4 Assumptions

1. PhonePark-equipped vehicles are uniformly distributed among all vehicles, thus a parking (or deparking) activity is performed by a PhonePark-equipped vehicle with probability *b*.
2. Parking activities, if detected, are detected independently of each other. In other words, given any two parking activities A and B such that each is performed by a PhonePark-equipped vehicle, whether A is detected or not is independent of whether B is detected or not.
3. $a_1(t), a_2(t),..., a_m(t)$ are independently and identically distributed. Particularly, they have a common mean which is denoted by $q(t)$.

Furthermore, we consider the situation in which only the Bluetooth connectivity method is used for parking availability. In this situation, false positive errors are negligible. Thus, we assume that *fp*=0. Now we show that $\hat{a}_i(t)$ is an unbiased estimate of $a_i(t)$. We treat $\hat{a}_i(t)$ as a random variable here since it depends on the parking and deparking detections which involve random errors.

Lemma 1: The conditional expectation of $\hat{a}_i(t)$ given $a_i(t)$ is equal to $a_i(t)$, i.e., $E(\hat{a}_i(t) \mid a_i(t)) = a_i(t)$.

Proof: Observe that since each parking activity is detected by PhonePark with probability $b\cdot(1-fn)$ and parking activities are detected independently of each other, $p_i(t)$ follows a Binomial distribution with $n=P_i(t)$ and $p=b\cdot(1-fn)$. That is,

$$p_i(t)|P_i(t)\sim\text{Binomial}(P_i(t),\ b\cdot(1-fn)) \tag{4.3}$$

Similarly,

$$d_i(t)|D_i(t)\sim\text{Binomial}(D_i(t),\ b\cdot(1-fn)) \tag{4.4}$$

When $fp=0$, according to the HAP algorithm, it follows that

$$\hat{a}_i(t)\,|\,a_i(t) = N-(p_i(t)\,|\,P_i(t))\cdot\frac{1}{b\cdot(1-fn)}+ (d_i(t)\,|\,D_i(t))\cdot\frac{1}{b\cdot(1-fn)} \tag{4.5}$$

Thus,

$$E(\hat{a}_i(t)\,|\,a_i(t)) = N-E(p_i(t)\,|\,P_i(t))\cdot\frac{1}{b\cdot(1-fn)}+ E(d_i(t)\,|\,D_i(t))\cdot\frac{1}{b\cdot(1-fn)} \tag{4.6}$$

According to Eq. 4.3 and Eq. 4.4, it follows that

$$E(p_i(t)\,|\,P_i(t)) = P_i(t)\cdot b\cdot(1-fn) \tag{4.7}$$

$$E(d_i(t)\,|\,D_i(t)) = D_i(t)\cdot b\cdot(1-fn) \tag{4.8}$$

Thus,

$$E(\hat{a}_i(t)\,|\,a_i(t)) = N-P_i(t)+D_i(t)=a_i(t) \tag{4.9}$$

Q.E.D.

Lemma 2. $E(\hat{a}_i(t))=q(t)$ for $i=1,2,\ldots,m$.

Proof:

$$E(\hat{a}_i(t)) = \sum_{k=0}^{N} (prob\ \{a_i(t) = k\} \cdot E(\hat{a}_i(t) \mid a_i(t) = k)$$

$$= \sum_{k=0}^{N} (prob\{a_i(t) = k\} \cdot k)$$

$$= E(a_i(t))$$

$$= q(t) \qquad (4.10)$$

Q.E.D.

Now we study the relationship between $\hat{q}(t)$ computed by Equation 4.1 and the true mean $q(t)$. Since $a_1(t), a_2(t), ..., a_m(t)$ are independently and identically distributed, $\hat{a}_1(t), \hat{a}_2(t), ..., \hat{a}_m(t)$ are also independently and identically distributed. Recall that

$$\hat{q}(t) = \frac{\sum_{i=1}^{m} \hat{a}_i(t)}{m} \qquad (4.11)$$

According to the central limit theorem, $\hat{q}(t)$ approximately follows a normal distribution with mean equal to $E(\hat{a}_i(t))$ and variance equal to $\frac{\operatorname{var}(\hat{a}_i(t))}{m}$. According to Lemma 2, $E(\hat{a}_i(t)) = q(t)$. $\operatorname{var}(\hat{a}_i(t))$ is set to be $\hat{Q}(t)$ computed by Equation 4.2.

Now we can talk about confidence towards estimating $q(t)$. That is, given an error tolerance δ, with what probability the difference between $\hat{q}(t)$ and $q(t)$ is smaller than δ. Let $\lambda = \delta \cdot \sqrt{\frac{m}{\operatorname{var}(\hat{a}_i(t))}}$. According to the property of normal distribution,

$$
\begin{aligned}
&\operatorname{Prob}\{|\hat{q}(t) - q(t)| < \lambda \cdot \sqrt{\frac{\operatorname{var}(\hat{a}_i(t))}{m}}\} \\
&= 2 \cdot \Phi(\lambda) - 1 \\
&= 2 \cdot \Phi(\delta \cdot \sqrt{\frac{m}{\operatorname{var}(\hat{a}_i(t))}}) - 1 \\
&= 2 \cdot \Phi(\delta \cdot \sqrt{\frac{m}{\hat{Q}(t)}}) - 1 \qquad (4.12)
\end{aligned}
$$

Equation 4.12 quantifies the following intuitions: i) the more samples are collected (i.e., the bigger m), the higher confidence; ii) the lower requirement on reliability (i.e., the bigger δ, the higher the confidence; iii) the higher fluctuation of the estimated availability (i.e., the bigger $\hat{Q}(t)$), the lower the confidence.

Example: Suppose that the standard deviation of $\hat{a}_i(8:00am)$ is 10 (i.e., the average fluctuation of *estimated* availability at the 8:00am is 10). Now if we want the error of the estimated mean $\hat{q}(t)$ and the true mean q(t) to be sm*a*ller than *2* with 90% confidence, then we need *to* collect 68 permitted periods. This amounts to about two months if each day is a permitted period.

7.4 Parking Availability Estimation algorithms

In this section we present four algorithms for estimating parking availability in real-time. Denote by $x(t)$ a random variable representing the true parking availability in S at time t of a permitted period. Denote by $\hat{x}(t)$ the parking availability estimated for time t of the same permitted period.

7.4.1 Historical Statistics (HS)

In HS, the real-time parking availability at time t of a permitted period is taken to be the estimated historical mean $q(t)$, i.e.,

$$\hat{x}(t) = \hat{q}(t)$$

7.4.2 Scaled PhonePark (SPP)

In SPP, PSD-observed parking availability is scaled to the entire set of parking spaces in a street block based on the penetration ratio and detection accuracy. Specifically, in SPP, the estimation of parking availability is initialized to be N at the beginning of a permitted period. Then when a parking report is received, the parking availability is decreased by $\frac{1-fp}{b\cdot(1-fn)}$ and set to 0 if it becomes negative after the decrease; when a deparking report is received, the parking availability is increased by $\frac{1-fp}{b\cdot(1-fn)}$ and set to N if it exceeds N.

7.4.3 Weighted Average (WA)

This algorithm computes a weighted average between the historical mean and the scaled PSD-observed parking availability. Specifically, let $\hat{a}(t)$ be the parking availability estimated by SPP. Then

$$\hat{x}(t) = w_{HS}\hat{q}(t) + (1 - w_{HS})\hat{a}(t) \qquad (5.1)$$

The weight w_{HS} is determined by experiments as will be discussed in Section 7.6.1.

7.4.4 Kalman Filter (KF)

We make the following two hypotheses:

Hypothesis 1. The true parking availability is equal to the historical mean plus a Gaussian noise with mean zero and variance equal to the historical variance, i.e.,

$$x(t) = q(t) + w(t) \quad (5.2)$$

where w(t) is a random variable normally distributed with mean zero and variance Q(t). w(t) is called the state noise.

Hypothesis 2. The scaled PSD-observed parking availability (i.e., the availability estimated by SPP) is equal to the true parking availability plus a Gaussian noise, i.e.,

$$\hat{a}(t) = x(t) + v(t) \quad (5.3)$$

where *v*(*t*) is a random variable normally distributed with mean zero and variance *R*(*t*); *R*(*t*) are to be estimated. The mean of v(t) is zero because the expectation of $\hat{a}(t)$ is equal to x(t) according to Lemma 1. *v*(*t*) is called the observation noise.

Using Kalman filter, and replacing q(t) by $\hat{q}(t)$, *x*(*t*) is estimated as follows:

observed residual: $y(t) = \hat{a}(t) - \hat{q}(t)$ (5.4)

Kalman gain: $K(t) = \dfrac{Q(t)}{Q(t) + R(t)}$ (5.5)

state estimate: $\hat{x}(t) = \hat{q}(t) + K(t) \cdot y(t)$ (5.6)

Incorporating (5.4) and (5.5), Eq. (5.6) can be rewritten as

$$\hat{x}(t) = \frac{R(t)}{Q(t)+R(t)} \cdot \hat{q}(t) + \frac{Q(t)}{Q(t)+R(t)} \cdot \hat{a}(t) \qquad (5.7)$$

From Eq. 5.7 it can be seen that the estimated parking availability is a weighted average between the PSD-observed parking availability (i.e., $\hat{a}(t)$) and the historical mean (i.e., $\hat{q}(t)$). The weights are inversely proportional to the expected errors. Specifically, Eq. 5.7 has the following properties:

1. The bigger the historical variance (i.e., Q(t)), the heavier is a(t) weighted towards the final estimate. This property quantifies the intuition that PhonePark is more useful when the true availability fluctuates in a wide range from the historical mean. In this case the parking availability is highly unpredictable and thus real-time observations are more important.

2. The bigger the value of $R(t)$, the heavier $\hat{q}(t)$ is weighted. This property quantifies the intuition that the historical statistics are more useful when real-time observations are inaccurate.

The observation noise parameter R(t) is estimated by the method introduced in [89] called ALMF (adaptive, limited memory filter).

7.5 Evaluation by Simulations

Our simulations used real world parking availability data retrieved from SFPark.org. SFPark.org provides APIs for access to real-time parking availability for street blocks in the city of San Francisco. Using these APIs we built a program that retrieves parking availability of street blocks every second. The simulations use SFPark data collected in a four month period from April 10 to August 11, 2012.

The experiments were conducted on two street blocks in San Francisco, namely 300-398 Polk Street and 2050-2098 Chestnut Street (see Figure 36). The Polk block has 12 parking slots in total and 4.6 available slots on average. The Chestnut block has 4 parking slots in total and 1.5 available slots on average.

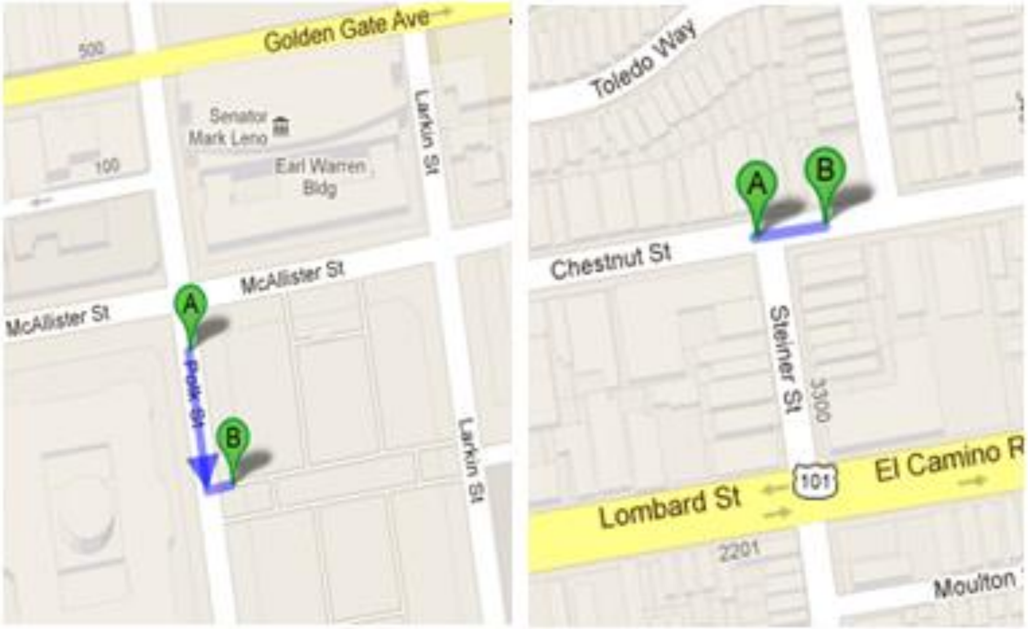


Figure 37-Street blocks considered for PAE evaluation

For a considered street block, we derive a sequence of PSD reports from its SFPark data, using the following procedure. We sequentially visit each change of parking availability recorded in the SFPark data. Let t_{last} be the time of the last visited change and t_{cur} be the time of the currently visited change. If the change indicates that some parking slots have been released since t_{last}, then each of these parking slots is released by a PhonePark driver with probability b. Furthermore, a release executed by a PhonePark driver is detected by PhonePark with probability $1–fn$. Thus, each release since t_{last} is detected by PhonePark with probability $b·(1–fn)$. Thus, for each release since t_{last}, a deparking report is generated with probability $b·(1–fn)$, at time t_{cur}. Let r be the number of generated reports for the currently visited change.

In addition, we generate a random number of extra deparking reports which simulate false positive errors. The random number follows a negative Binomial distribution NB(r, fp)[3]. The justification is that, by its definition, NB(r, fp) represents the number of false positive reports that are supposed to be generated until r correct reports are generated, given that the probability for a report to be false positive is fp. A similar procedure follows for the case that the change at t_{cur} indicates that some parking slots have been occupied since t_{last}. Now we are ready to evaluate various parking availability estimation algorithms. We use 10-fold cross validation. For this we split the SFPark data of a street block into pieces such that each piece covers exactly one permitted period. These pieces are then used for sampling of cross validation. All the system parameters and their values are shown in Table 12.

For the evaluation of the HAP algorithm the performance measure is the root mean square error (RMSE) between the estimated mean computed by HAP and the true mean. For the evaluation of the PAE algorithms, we consider two performance measures:

1. The *RMSE* between the estimated average availability and the true average availability;
2. The *boolean availability accuracy* which is defined as follows. We say that a PAE algorithm is *boolean correct* at time t if (i) the algorithm indicates that there is at least one slot available and there indeed is, or (ii) the algorithm indicates that there is not any slot available and there indeed is not. The boolean availability accuracy is defined to be the percentage of the number of time units at which a PAE algorithm is boolean correct.

---

[3] In simulations fp is fixed for each experiment.

Table 12 - System parameters and their values.

Parameter	Symbol	Value
Penetration ratio	*b*	1%, 50%
False negative probability	*fn*	0, 0.05, 0.1, 0.15, 0.2, 0.25
False positive probability	*fp*	0, 0.05, 0.1, 0.15, 0.2, 0.25
Length of collection period		4 months
Length of permitted period		1 day
Length of time unit		minute

7.6 Evaluation of HAP

In this subsection, we evaluate the performance of HAP on computing historic parking availability profiles for a street block. Each experiment is conducted as follows. For each permitted period, both the historic mean and HAP's estimation is recorded. For each time unit within the permitted period, the mean of both HAP's estimation and the historic mean are computed across the permitted periods. The RMSE is then computed across all the time units. Each experiment is repeated several times and the average RMSE taken.

Figures 37-40 show the RMSE of HAP From Figure 37 it can be seen that for the Polk block, the RMSE of HAP is below 2.3 when the penetration ratio is 1%. On the other hand, the average true availability is 4.6. A higher penetration ratio or a longer collection of historical observations can increase accuracy. For example, Figure 38 shows that when the penetration ratio increases to 50%, the RMSE of HAP is below 0.72, which is 16% of the average true availability of the Polk block. For the Chestnut block, the RMSE is below 0.57 when the penetration ratio is 1% and below 0.47 when the penetration ratio is 50%, while the average true availability is 1.5 (see Figures 39 and 40). Also, observe that as the false positive or false negative probability increases, the algorithm generally becomes less effective.

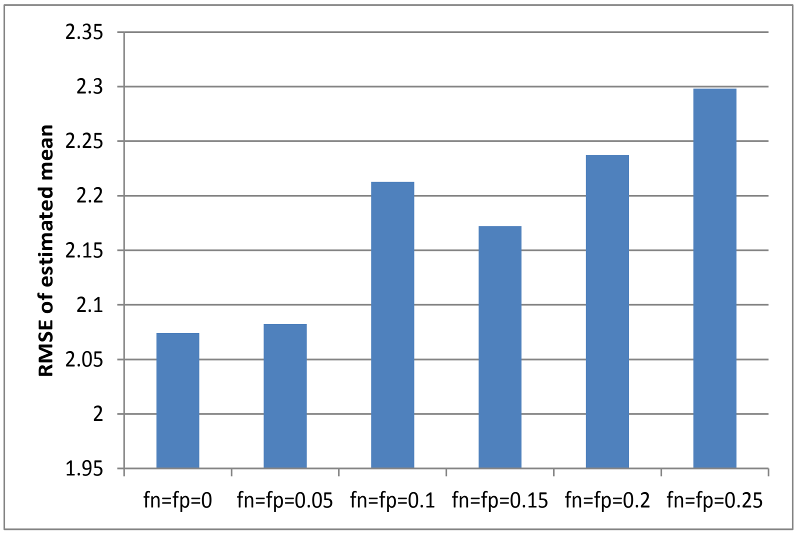


Figure 38-RMSE of HAP, b=1%, Polk St.

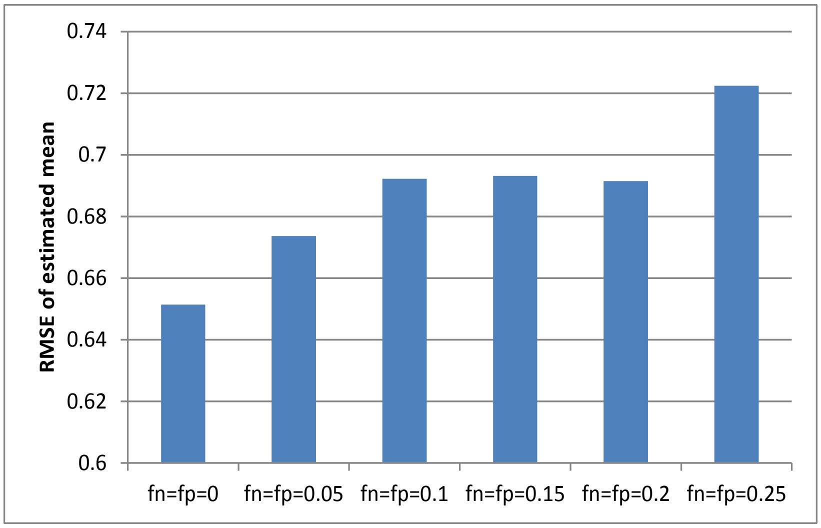


Figure 39-RMSE of HAP, b=50%, Polk St.

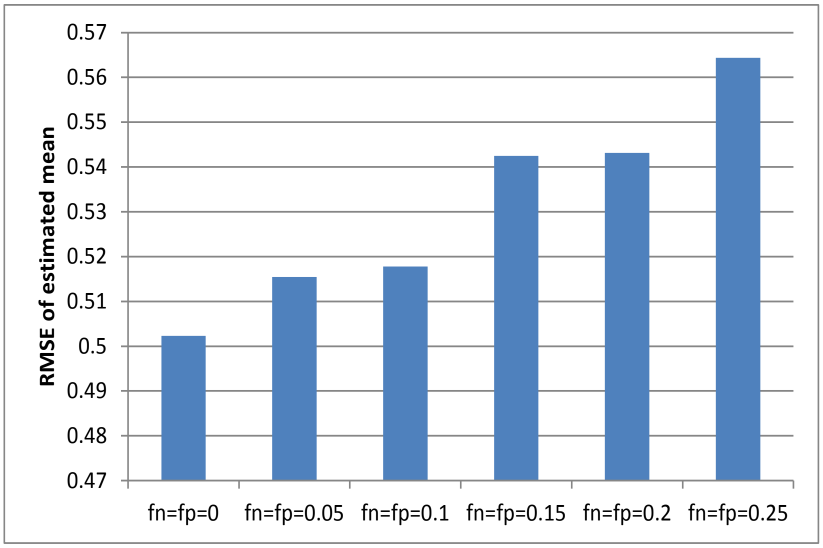


Figure 40-RMSE of HAP, b=1%, Chestnut St.

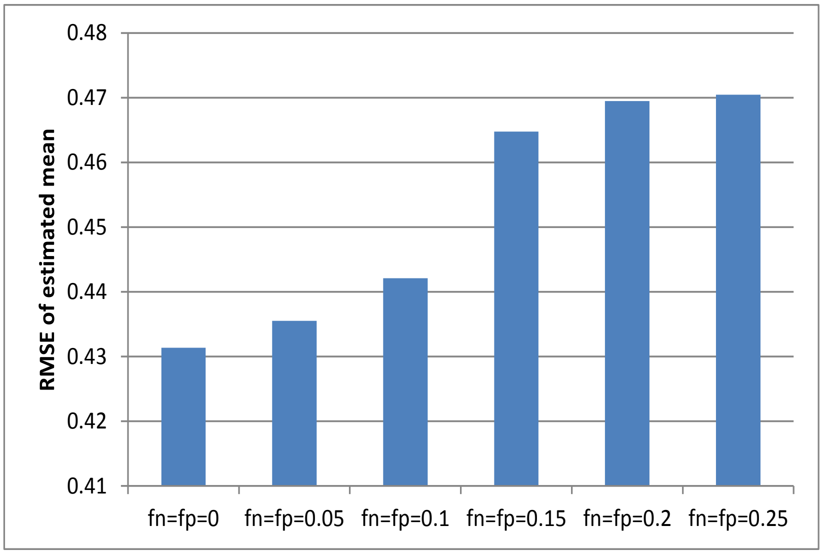


Figure 41-RMSE of HAP, b=50%, Chestnut St.

7.6.1 Tuning-up Weighted-Average (WA)

Experiments are conducted to determine the optimal weight w_{HS} for the WA method. We vary w_{HS} between 0 and 1 with a step size of 0.1. Figure 41 shows that different weights produce . HAP's estimation of $q(t)$ is done using the same penetration ratio as in computing $\hat{a}(t)$ for all the configurations in Figure 41. For example, if 5% penetration ratio is considered for estimating $q(t)$ by HAP, then 5% is also utilized for $\hat{a}(t)$ estimation.

From Figure 41, it can be seen that the optimal weight, i.e., the weight that leads to minimum RMSE, depends on parameters including the penetration ratio *b*, the false positive probability *fp*, the false negative probability *fn*. For example, the optimal weight is 0 when *b*=1%, *fn*=*fp*=0.1 for the Chestnut block whereas it is 0.6 when *b*=50%, *fn*=*fn*=0.25. Figure 41 demonstrates that the optimal weight for the WA method can be calibrated.

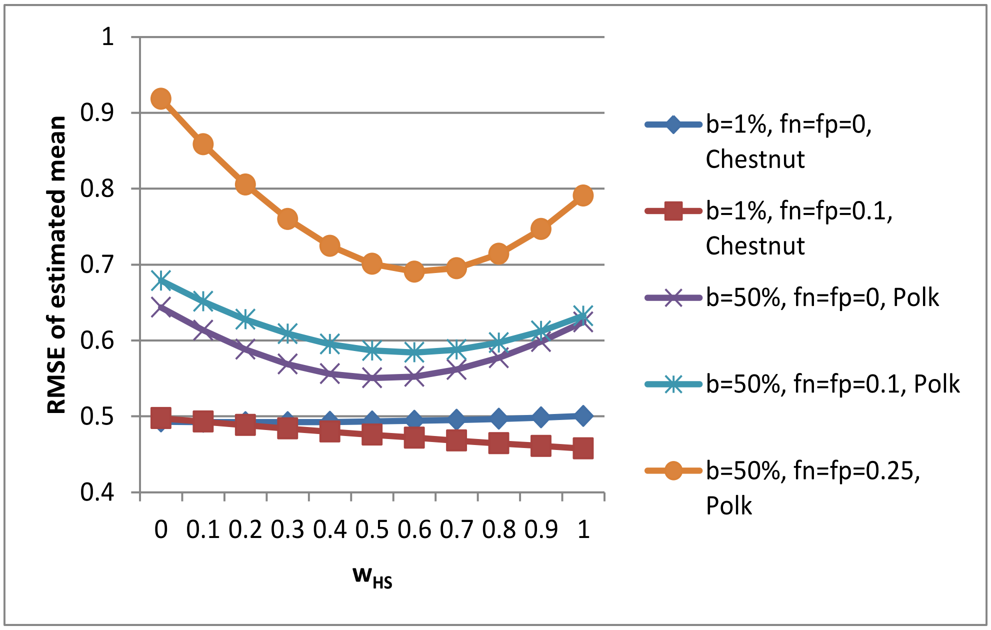


Figure 42-Determining weight (WHS) for WA

7.7 Comparison of HS, SPP, WA, and KF

In this section, we present evaluation results for the four algorithms for estimating street parking availability in real time. The four parking availability estimation algorithms are (1) Scaled PhonePark (SPP), (2) weighted average (WA), historical statistics (HS), and (4) Kalman Filter (KF). For the WA approach, the optimal weight is determined as discussed in Section 7.6.1.

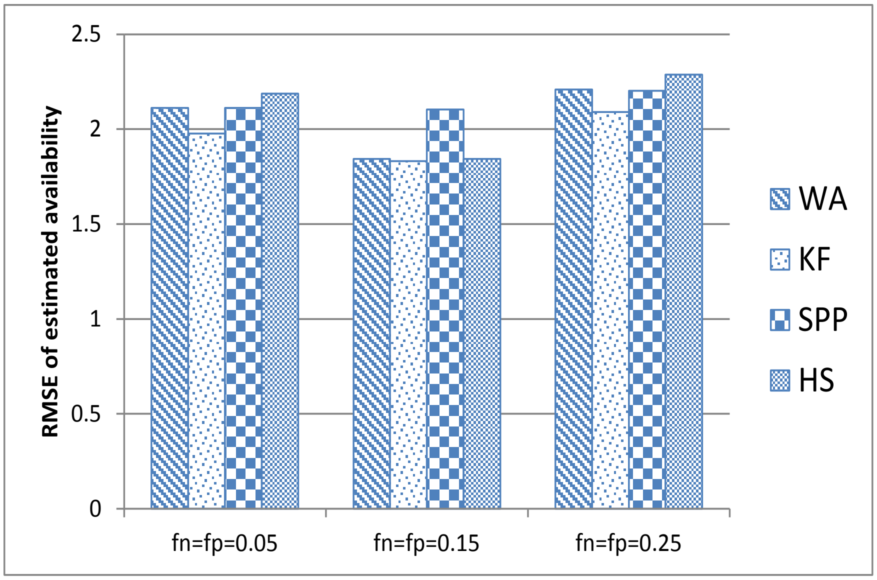


Figure 43-RMSE of PAE algorithms, b=1%, Polk St.

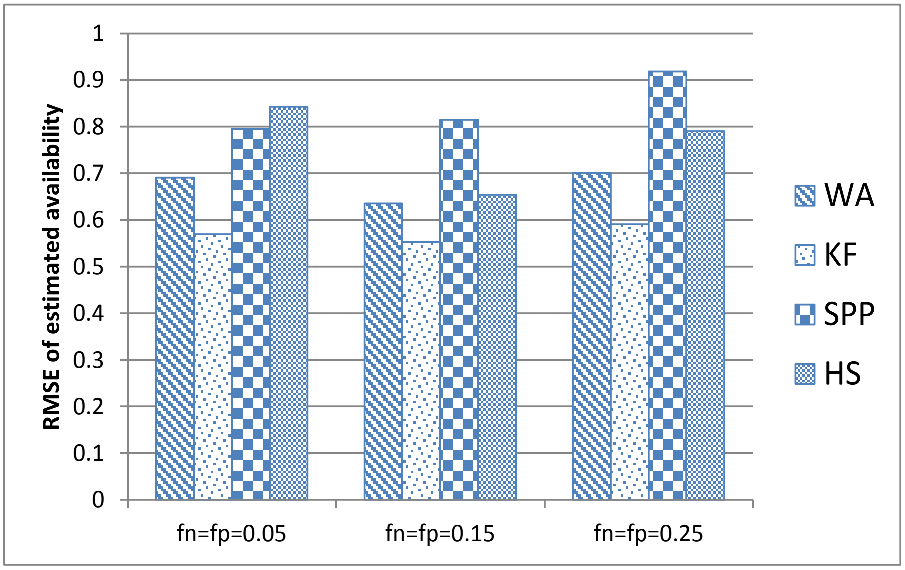


Figure 44-RMSE of PAE algorithms, b=50%, Polk St.

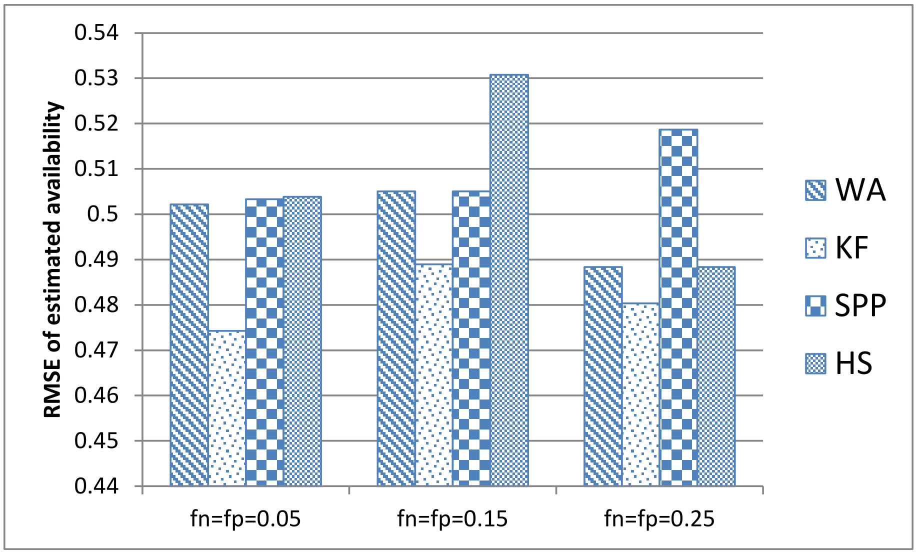


Figure 45-RMSE of PAE algorithms, b=1%, Chestnut St.

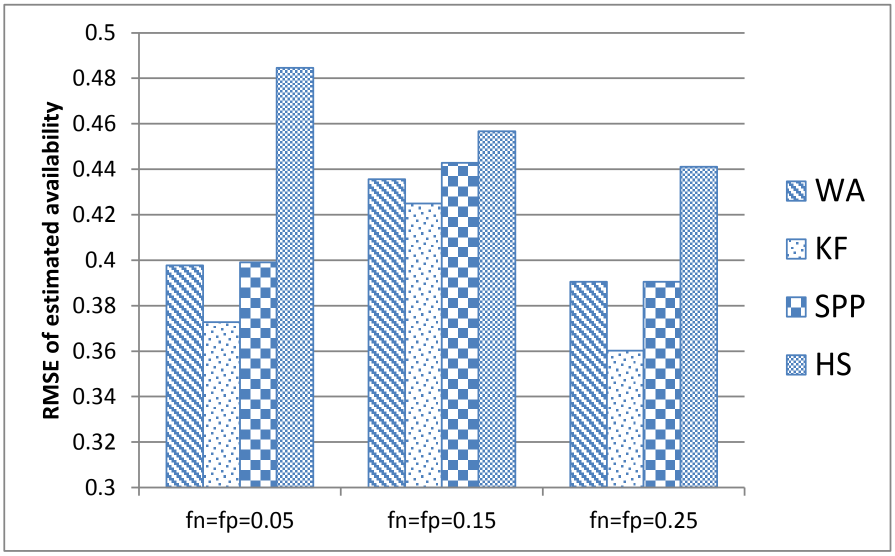


Figure 46 - RMSE of PAE algorithms, b=50%, Chestnut St.

7.8 Comparison of RMSE

Figures 42-45 show the RMSE performance of the PAE algorithms for different penetration ratios and different street blocks. From the figures it can be seen that in most cases

KF has the lowest RMSE among all the algorithms. Furthermore, for all PAE algorithms, the RMSE decreases as the penetration ratio increases. Particularly, when the penetration ratio is 1%, the RMSE of KF for Polk St. is about 2 (see Figure 42), which is 40% of the average parking availability of the street block. When the penetration ratio increases to 50%, the RMSE of KF reduces to about 0.6 (see Figure 43), which is 13% of the average parking availability. For the Chestnut St., the RMSE of KF is 30% of the average parking availability when the penetration ratio is 1% (see Figure 44).

In most cases, WA is second to KF in terms of the RMSE performance. This indicates that dynamic weighting between PSD-observed parking availability and the historical mean, as done in KF, works better than static weighting as done in WA. In addition, the accuracy of WA is higher than that of SPP and HS. This shows the benefit tuning up w_{HS} for WA.

7.9 Comparison on Boolean availability accuracy

Figures 46-47 show the Boolean availability accuracy of the algorithms. From the figures it can be seen that WA has the highest accuracy among all the algorithms. For the Polk St., the accuracy of WA reaches 90% even when the penetration ratio is only 1% (see Figure 46). For the Chestnut St., the accuracy of WA reaches 70-75% when the penetration ratio is 1% (see Figure 47). For both blocks, HS is second to WA. Notice that even though KF is better than WA on RMSE, it is not effective as WA on boolean availability accuracy.

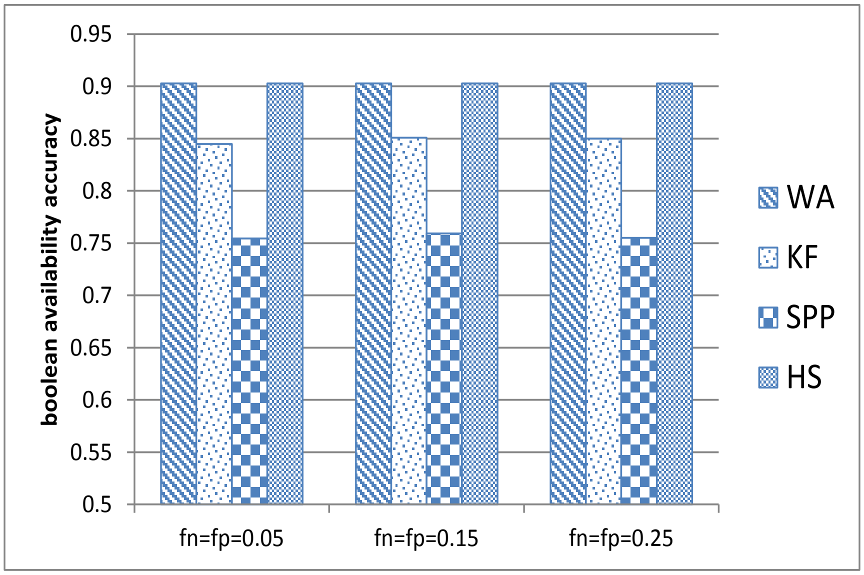


Figure 47-Boolean availability accuracy, b=1%, Polk St.

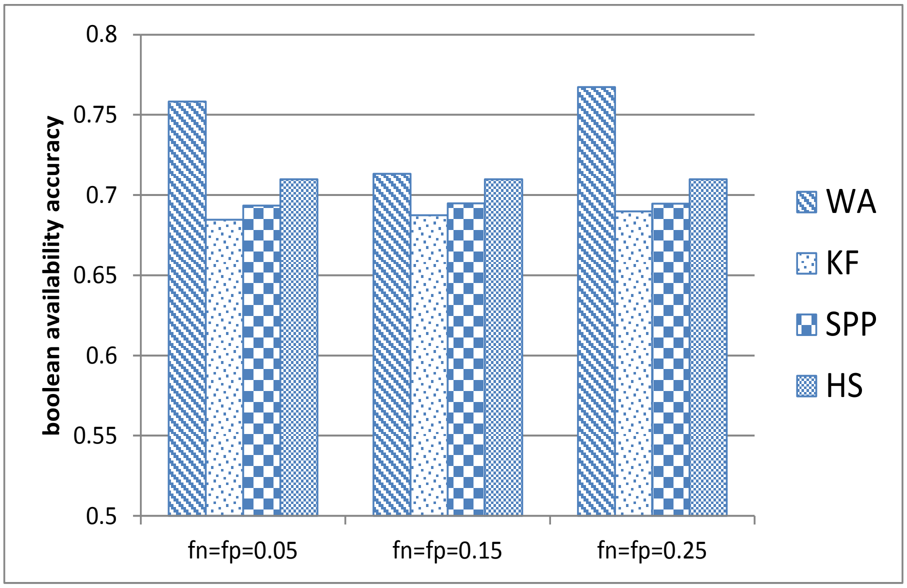


Figure 48-Boolean availability accuracy, b=1%, Chestnut St.

7.10 Relevant work

A number of approaches have been considered for monitoring parking spaces. Some approaches monitor parking spaces by color histogram classification and car feature point detection from still images captured by externally mounted cameras [44, 45]. Our solution only needs sensors on a mobile phone and not external cameras.

Parking garages use in-out counters at exit and entry points to compute the number of additional vehicles they can accommodate [45]. There exist other systems that allow travelers to access parking information and make prior reservations for parking in areas such as airports and rail stations [46, 47]. PhonePark is different and focuses on street parking.

The GPS sensor has been used in the past for parking guidance. For example, GPS guided parking is used in [48]. PhonePark solves a different problem, namely automatically detecting parking and deparking activities. The available parking spaces detected by PhonePark may serve as an input to the parking guidance system.

The availability of the vacant parking spaces can be calculated by external sensors such as those installed in the parking areas, which count the number of cars which enter and exit from the parking space [17]. For example, in San Francisco, several parking projects have been initiated [17, 18] that utilize externally implanted sensors. However, this strategy is expensive to deploy and maintain. For example, in one project that covers 8000 parking spaces the cost was over USD $23 million [17, 19]. The proposed work in this dissertation is not dependent on external sensors.

Ultrasonic sensors at the top of each parking space or on vehicle side-doors [19] can be used to sense the availability or unavailability of each parking space. Our solution uses only sensors available on mobile phone to infer when and where a traveler had parked their car.

Sensors implanted under road surfaces or attached to the car side-doors are expensive to deploy and maintain (e.g., [17] cost USD $500 per system for each parking space, and [19] cost USD $400 per system for each car). These sensors may underperform in extreme weather. For example, in heavy snow these sensors may be covered. Using mobile phones is cheaper, more convenient, and more flexible.

The works [49, 50, 51] are based on parking but the focus is on how to choose parking slots ideally in competitive parking settings. Thus, in these other works routing to the most optimal parking slot is the focus. This dissertation is different and focuses on estimating the current street parking availability on a street block in real time.

The work in [52, 53] analyze the capacity of an opportunistic system to assist the search for parking space. Vehicles searching for parking space are equipped with sensors that allow them to sense the location and status of parking spots as they drive across the city. This information is subsequently shared upon encounters with other vehicles or submitted to a central server. The proposed work in this dissertation is different and focuses on detecting and estimating the availability of street parking spaces. On the other hand, [52, 53] focus on optimizing the effectiveness of the parking search process through user oriented performance metrics, such as the parking search time, route length, and the proximity of the found or assigned parking spot to the user travel destination.

Other areas of parking research includes pursuing parking spaces and responding to pricing policies about public and private parking facilities that is studied in [54, 55]. The drivers are modeled as strategic agents who make rational decisions while minimizing the cost of the acquiring parking spots, under deterministic or probabilistic information for the overall parking demand.

7.11 Discussion

In this section, historical parking profile construction and parking estimation algorithms are proposed. The algorithms are validated using real-time and real world parking availability data. For parking status detection, we propose a cost effective solution that utilizes mobile phone sensors such as GPS, accelerometer, and Bluetooth sensors. Furthermore, the parking status detection algorithm may piggy back on pay-by-phone for parking transactions.

Given the fact that not all drivers carry a mobile phone and the fact that not all drivers with a mobile phone have the PhonePark system installed, the penetration ratio of the PhonePark system may not be 100%. Additionally, PhonePark PSDs may produce false observations of parking activities due to GPS errors, transportation mode detection errors, and Bluetooth pairing errors. The parking availability estimation algorithms take these limitations into consideration and then provide a final estimate on the availability of street parking spaces.

Specifically, parking availability estimation (PAE) algorithms estimate the street parking availability in real-time by combining historical parking statistics with real-time parking observations. Several approaches on combining historical parking statistics with real-time parking observations are considered. Experimental and theoretical analysis demonstrate the effectiveness of the parking availability estimation algorithms. Specifically, for penetration ratios of 1%, the average error of street parking availability estimation is below 2. More studies show that the combination of historical parking statistics and real-time parking observations as done in weighted average (WA) and Kalman Filter (KF) is more effective than using solely historical parking statistics (HS) or solely real time observations (SPP).

In general, the proposed approach is economical, convenient, and flexible. It produces good estimation in real-time on the availability of street parking spaces. The estimation of parking availability has potential to improve drivers’ parking experience by saving parking search time, reducing gas consumption and CO_2 emission.

8. PRIVACY AWARE MOBILE SENSING

A GPS sensor is available on many smart phones that can provide within 20m accuracy [58, 59, 60, 61, 62]. Many third party location based services are been created randomly to provide services such as location based social networks, transportation itinerary planning, and location based coupons/deals. In these systems, the mobile users reveal their locations. If these locations are not adequately protected, it may reveal a person's political views, religious affiliations, or state of health. Knowledge of a mobile user’s location may lead to stalking or unwanted advertisements sent to the mobile device with the marketing of products or services [91, 92, 93]. There has also been an increase in the number of GPS based harassments [5].

Location privacy preservation aims to prevent adversaries from learning a mobile user’s past or current locations, or the times of the visits. To preserve location privacy, Gruteser and Grunwald [59] introduced the K-anonymity model in location context. In this model, location privacy ensures that a mobile user’s location is indistinguishable from K-1 other user locations. The K value denotes the desired minimum anonymity level. A value of K = 1 implies that anonymity is not required for the message. A value of K > 1 means that the message will be assigned a spatiotemporal cloaking box that is indistinguishable from at least K-1 other messages, each from a different mobile client. Therefore, larger K values imply higher degrees of privacy. One way to determine the appropriate K value is to assess the certainty with which an adversary can associate the message. This certainty is given by 1/K.

Another level of protection demanded in location based systems is safety against request/query linking, by preventing an adversary from knowing the mobile user that has submitted a sensitive query. Sensitive queries such as, “*Where is closest XXX Rehab Center to my current location?*”

should not be linked to the mobile user. In query privacy, the location component inside the query is the quasi-identifier [94]. Fortunately, the concept of K-anonymity can also be used for query privacy protection. Mobile users in these frameworks are considered K-anonymous if a query cannot be distinguished from at least K-1 other queries. The technique is to expand the query location point until we locate K-1 other queries. This way, the exact query locations are hidden.

Traditional approaches such as encryption, hashing or removing the mobile user identification (e.g. user name) from the submitted request cannot overcome the privacy threats in location based systems because the quasi identifier is location and not user-id. Furthermore, encryption provides an “all (100%) or nothing (0%)” service. In these privacy aware systems, mobile users may want a personalized percentage of privacy (e.g. 80%), this cannot be done using encryption. Moreover, location based log files may be released to the public for research or data mining purposes. Due to the uncertainty that would have been introduced by encryption, this released data would be difficult to mine.

Personalized control of location information disclosure is needed in location based systems because the privacy requirement varies from one mobile user to another. For example, in a location based social network application such as Foursquare, some people may feel comfortable revealing their exact location at all times, other mobile users may feel comfortable revealing the zip code only, or the city, or country. Clearly, mobile users may need a model that allow them to define the personalize privacy level that they desire.

Several privacy focus architectures are considered for privacy aware location based services. These architectures are *client-server*, *trusted third party*, and *peer based*. In the strict client server architecture, clients communicate directly with the LBS by submitting a request to the LBS, the LBS then returns a response directly to the client [61, 62, 63, 64]. In the peer based model, clients communicate directly with each other to achieve location privacy [10]. The intention of the clients is to cloak with each other in order to satisfy location K-anonymity. The trusted third party model utilizes the concept of a middle-ware between the mobile user and the LBS. We sometimes refer to the middle-ware as an *anonymization server* or *AS*. Mobile requests are first sent to the middle-ware, the incoming request is then cloaked with other requests by the anonymization server before submission to the LBS. The proposed work is focused on the trusted third party architecture since these systems are now being deployed to the public [93]. The trusted third party framework is also common in privacy preserving location based systems [6, 9, 22, 60, 66].

The proposed privacy solution is effective for both snapshot and continuous query systems. A snapshot query is a request submitted once by the mobile user. For example, "*Where is the closest sushi restaurant?*" On the other hand, a continuous query is submitted at discrete time points by the same mobile user. For example, "*Continuously send me gas price coupons as I travel the inter-state highway?*" Most of the current work have focused on snapshot queries. However, since the cloaking set for the same mobile user may be different at discrete time stamps, a snapshot solution may not be sufficient in a continuous querying environment as indicated by [66].

Importantly, we observe that current anonymization techniques [60, 22, 95] underperform if K-1 other users or requests cannot be found. Consequently, the request is discarded since it cannot be anonymized. One contribution of the proposed work is ensuring that all incoming location based requests can be safely anonymized to the desired level.

8.2 Preliminaries

To define the problem of privacy preservation in mobile location based services, first, we need to define the privacy information that we are interested in preserving. Second, the threat model or background knowledge that an adversary may use to attack the privacy of users in a location based system. Variations in formulation of these issues may lead to different versions of privacy preservation in mobile location based systems. In this dissertation, we discuss a version which we believe is useful in many location based systems.

In this section, we will present formal definitions, requirements, architecture, threat model, and adversaries' knowledge.

8.2.1 Formal definitions

Definition 9 (LBS Quasi-Identifier) A set of attributes {q1, . ., qn} of a mobile query is called a LBS quasi-identifier if these request parameters can be linked with external data to uniquely identify at least one mobile client in the system.

One example of quasi identifier (QI) in location based systems is the location {*latitude, longitude*} of the mobile user submitting the request. The definition recalls the definition of QI introduced in relational databases [96].

Definition 10 (Location-based snapshot query) A location-based snapshot query $Q_t^{snapshot}$ is a location based query submitted at time t. Such a query will not be submitted at time *t+n,* $\forall n > 1, t \geq 0$.

Definition 11 (Location-based continuous query) A location-based continuous query $Q_t^{continuous}$ is a sequence of snapshot queries submitted at discrete time points. $Q_t^{continuous} = \{ Q_t^{snapshot}, Q_{t+1}^{snapshot}, Q_{t+2}^{snapshot}, Q_{n-1}^{snapshot}, Q_n^{snapshot} \}$. For example, "*Continuously send me information on public passenger buses that are within five minutes from my current location?*"

Definition 12 (Region request) A region request R_t is a group of queries that are anonymized together to satisfy the K-anonymity requirement, an area encompassing all the queries is submitted to the LBS service provider. Each region request $\mathfrak{R}$ is formalized as:

$$\mathrm{Rt} = (\mathfrak{R}_{\mathrm{id}}, \mathbb{Q}_{\mathrm{set}}, \mathcal{L}_{\mathrm{t}})$$

where $\mathfrak{R}_{\mathrm{id}}$ is the identifier of the region request, $\mathbb{Q}_{\mathrm{set}}$ is the set of queries contained in Rt and $\mathcal{L}_{\mathrm{t}} = (\ell^{\mathrm{t}}_{\mathrm{x-}},\ \ell^{\mathrm{t}}_{\mathrm{y-}}, \ell^{\mathrm{t}}_{\mathrm{x+}}, \ell^{\mathrm{t}}_{\mathrm{y+}})$ is the location of the bounding rectangle for $\{\mathbb{Q}_1, \mathbb{Q}_2, .. \mathbb{Q}_{\mathrm{n-1}}, \mathbb{Q}_{\mathrm{n}}\}$ at time t.

Definition 13 (Local K-anonymity) A region request (R_t) satisfies local K-anonymity (K_{local}) if it contains at least K_{local} different location based requests.

Definition 14 (Global K-anonymity) A continuous query $Q_t^{continuous}$ satisfies global K-anonymity if the number of requests in intersection of the region requests or snapshots in the continuous query is at least K_{global}. Therefore, for n region requests R_1, R_2 ... Rn in the continuous query $|R_1 \cap R_2 \cap \ ... Rn\ | >= K_{global}$ and $K_{global} <= K_{local}$.

8.3 Requirements

- When a mobile user submits a location based request, she should be given the option to hide her location. The location is an identifier in location based services. Let $q_m(x, y)$ be a query q submitted by mobile user m from a point (x, y) where latitude = x and longitude = y. The request q_m is converted to some $q_{m'}((x1, x2), (y1,y2))$, where $q_{m'}$ is the new query and $(x, y) \in ((x1, x2), (y1,y2))$ such that $((x1, x2), (y1,y2))$ is a region encompassing the point (x, y).
- When a mobile user submits a sensitive query, the query should not be linkable to the mobile user. For example, a query such as "*Where is the closest parking lot to the HIV rehabilitation center?*" should not be linkable to the mobile user that submitted the

query. This kind of query is sensitive query and may be embarrassing to the sender if the query is revealed.

- Queries should not be discarded if K-1 other mobile clients or mobile requests cannot be found. In previous work [60, 22], if K-1 other mobile queries cannot be found, the query is discarded. Any privacy model that suffers from this pitfall is referred to as “best effort”. On the other hand, MobiPriv is a guaranteed service, and all queries can be anonymized successfully without a delay. Let *N* be the total number of mobile requests sent by mobile clients to the anonymization server. The number of mobile request that can be anonymized successfully by the anonymization server is M_t. The anonymization server ensures that:

$$\frac{|N|}{|M_t|} * 100 = 100\%$$

This implies that all the queries sent to the system will be anonymized successfully and none will be dropped. In our evaluations we refer to this property as the *success rate* of the privacy algorithms.

- Using historical data such as past queries should not influence query linking. This is frequent in continuous querying systems where the adversary can aggregate all the regions that he suspects a mobile client visits and use this information to link queries to the mobile client. Assume that mobile user u_k submits multiple requests with a privacy requirement of *K* from points in the regions R_1, R_2, R_3…R_n. Then,

$$R_1 \cap R_2 \cap R_3 \cap R_4 \cap \ldots R_n \neq u_k$$

- When a mobile user submits a request, if K-1 other requests are not available at the time of request, the system should not wait for these requests to become available.

Some location based services provide emergency and lifesaving functionalities, therefore delays cannot be tolerated.

8.4 System architecture

The system consists of mobile devices with positioning capabilities, location based services (LBS), wireless networks, and the proposed algorithms running on a privacy aware middle-ware called an anonymization server (AS). Below, we discuss each component. See Figure 1 for a layout of the architecture.

8.4.1 Mobile device (clients)

Mobile device includes mobile phone, PDA, and other devices such as laptops with positioning capabilities. First, each mobile device computes its physical location from the GPS or Wi-Fi component on the device. Mobile users specify the privacy requirement that they desire from the user interface of their mobile device in the proposed system. Both the personalized privacy requirement and the query containing the location data is forwarded to the anonymization server. Observe, for a typical query the location component of the query may not be the current mobile user's location. For example, Alice is at Point A and Alice submits a query requesting the closest buses to Point B. In this case, there is no need for Alice to reveal that she is at Point A.

8.4.2 Anonymization server (AS)

The anonymization server knows the location of all the mobile users. The physical location computed by the mobile device is sent to the anonymization server with the query. The role of the anonymization server is to privatize location and the request before submitting it to the location based system. We assume that anonymization server is the trusted third party, the adversaries loiter between the anonymization server and the location based system.

8.4.3 Secure communication service

The communication link between the mobile clients and the anonymization server is assumed to be wireless connections that are secure.

8.4.4 Service provider (LBS)

The service provider provides location based services to its subscribed mobile users. On receipt of a request from the anonymization server, the location based server processes the request and returns a response to the anonymization server. The service provider has the ability to process a given cloaked region and also process an exact point. The service provider is not responsible for privacy policies of the mobile clients.

8.4.5 Operation flow

Mobile users submit requests incorporating positioning information such as current location (latitude and longitude) as a parameter of the request to the AS. The AS then cloaks the client's query location point into a region containing *K-1* other mobile user request.

The AS then forwards the aggregate region request to the LBS. The LBS processes the query and sends a response. This response sent by the LBS is generic and should be filtered to get precise results. Filtering can be done on the AS or by the mobile client. Filtering on the mobile device may be costly due to limited battery power and processing capabilities. A diagram depicting the architecture is shown in Figure 48.

One novelty about our approach is, if *K-1* other mobile user request cannot be found we generate realistic diverse dummies instead of dropping the query as in previous anonymization approaches that uses the trusted middle-ware.

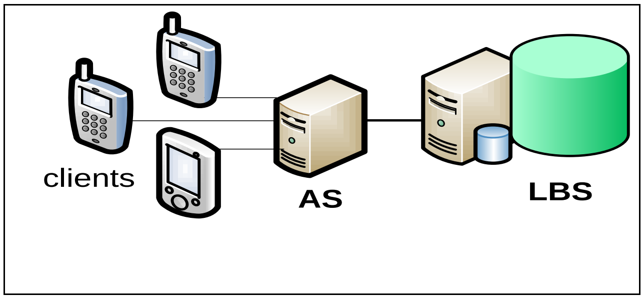


Figure 49-LBS with Anonymization Server (AS)

8.5 Threat model and adversary's knowledge

To evaluate the privacy protection of the proposed system, we consider the following threat model. Mobile users communicate precise personal location information, while adversaries' intention is to decipher the physical location of the mobile clients.

Additionally, as the mobile clients may submit sensitive queries, adversaries intend to infer which mobile user has submitted current or past queries. For example, if a person submits a query such as "Where is the less expensive bar in the red light district closest to Downtown Chicago?" Then, this query should not be linked to the sender.

In summary, the adversaries have two main goals: (1) Discovering the location that the submitted requests are related to, (2) Correlating a sensitive query to a mobile user. Adversaries loiter between anonymization server and location based server and is aware of the time that the mobile users may submit the queries.

Adversaries may "look up" the correlated street address corresponding to a mobile user location (latitude, longitude) or query location (latitude, longitude) using reverse geocoding techniques and they are aware of the location of some mobile users. Moreover, adversaries may observe the region sent by the anonymization server to the location based server and determine the number of queries in that region request. Furthermore, the adversary may have background knowledge about the victim. Background knowledge includes home address, office address, etc. Additionally, adversaries may perform query linking by observing the location component inside a query. Advance adversaries may also listen over time and correlate multiple queries sent by the same mobile users. These assumptions are realistic. Below we model some of the adversaries' capabilities and we will show that the proposed approach is resilient to these capabilities

8.5.1 Count (R)

Given a region R, with multiple mobile users or mobile requests $\{r_1, r_2, r_3 \ldots r_{k-1}, r_k\}$. The adversary can determine the number of queries in *R*. Thus, from R, the adversary can determine an integer value corresponding to $|\{r_1, r_2, r_3 \ldots r_{k-1}, r_k\}|$.

8.5.2 Lookup (lat, lon)

Given a location point (latitude, longitude), the adversary can perform reverse geocoding and determine the high level street address.

8.5.3 Intersect ($R_1, R_2, R_3, R_4 \ldots R_{n-1}, R_n$)

This capbility correlates multiple requests from the same sender. Given $(R_1, R_2, R_3, R_4 \ldots R_{n-1}, R_n)$, the adversary can perform $\{R_1 \cap R_2 \cap R3 \cap R4 \ldots R_{n-1} \cap R_n\}$ and determine common requests across regions.

8.6 Dummy requests

We define the term dummy request, or dummies, to be a *fake* mobile request U_a automatically and realistically generated by the system. Dummies should be generated in a way that an adversary cannot differentiate a dummy from a real user request U_r. In LBS systems such as road navigation systems, mobile users send continuous position queries. If dummies are generated randomly, then adversaries can easily find the difference between the real user and the dummies. For example, real requests tend to be related to road networks and dummies may wonder off into the Euclidian space if they are randomly generated. In our algorithm, the dummies are generated relative to the temporal and spatial property of the real user request. The identification numbers of the realistic dummies are taken from the dummy profile. The proposed dummies requests are also *diverse* and are different from the mobile user request.

This dissertation is not the first to use dummies as a concept to increase privacy in location based systems. However, it is the first to consider dummy request generation on the anonymization server (AS). *Kido et al* [62] introduced the concept of dummy location generation in location based systems but did not consider K-anonymity. In [62], the client sends the true position along with the dummy positions to the LBS. The LBS then responds with answers to both the true position and the dummy position. The proposed work is different as [62] did not consider the anonymization server instead only considered the client server architecture. We also considered dummy query generation and not dummy location generation as described in [62]. One disadvantage of the approach in [62] is the high processing cost on the mobile device to filter the results that were returned by the LBS. The excess filtering cost on the mobile client negatively affects the short battery life and limited processing power of hand-held mobile devices. Consequently, MobiPriv filters on the anonymization server.

Further, if an adversary has knowledge of history then a user may be re-identified in [62] by taking the intersection of the regions that the mobile user sent the requests from. Thus, [62] is not useful in a continuous query environment.

8.6.1 Dummy profile

In MobiPriv, the dummy profile is a file containing a list of all mobile users in the system along with corresponding dummy user identification numbers. We define *profileCount* to be the number of dummies associated with a real mobile user on the dummy profile. We initialize *profileCount* to be the maximum K value allowed in the system and construct the dummy profile as follows. For each possible real user U_r in the system, we associate a set of dummies with the real user identification. To generate dummy requests for a particular user, we first consult the dummy profile and take the dummies from the dummy profile in topological order. We generate dummy requests to satisfy the local anonymization groups which then become candidates to prevent query linking in continuous queries. The proposed work (i.e. MobiPriv) is the first to consider realistic diverse dummy user generation on the anonymization server.

We will now explain how to generate realistic dummy requests. Figure 49 shows a diagram of a cloaked request. The parameters *dx* and *dy* are the user defined spatial tolerances, and coordinates *x*, *y* represents the location (latitude and longitude) of the request. From *dx*, *dy*, *x*, and *y* we build the bounding box such that parameters *x1* and *x2* are the regions x-coordinates. Likewise, *y* and *y2* are the y-coordinates.

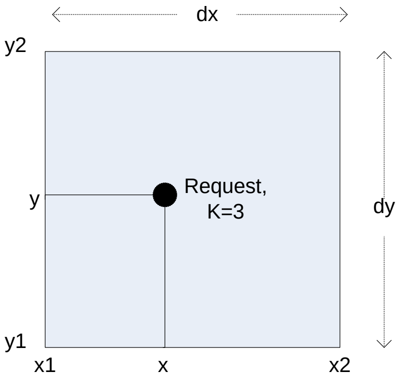


Figure 50-Dummy request generation

First, the following offsets are defined; (1) x_{offset}, and (2) y_{offset}. The values for these offset are computed based on the spatial properties (i.e. x and y) of the incoming request as shown below.

$$x_{offset} = \min{(x2 - x, x - x1)}/2$$

$$y_{offset} = \min{(y2 - y, y - y1)}/2$$

Let *<0,1>* means a random number between 0 and 1 inclusive. Given these offsets, we then use x + <0,1> * x_{offset} and y + <0,1> * y_{offset} as spatial properties for the dummy requests. Therefore, for each dummy request, the following is computed.

$$dummy_{offset}^{x} = \ x + < 0{,}1 > * \ x_{offset}$$

$$dummy_{offset}^{y} = \ y + < 0{,}1 > * \ y_{offset}$$

Where $dummy_{offset}^{x}$ is the dummy's x offset and $dummy_{offset}^{y}$ is the y offset. The temporal property of the dummy requests can be ascertained similarly. Using this strategy, the dummy requests are related to the spatial and temporal properties of the real request. Hence, the dummy requests may be perceived as real requests by adversaries.

The dummy generation algorithm is presented (see Algorithm 1). In line 4, *profileCount* is declared to be 50, likewise in the experimental evaluations. This implies that the system generates at most 50 realistic dummy requests to satisfy the privacy requirements. Line 6, the proposed work reads the dummy profile for the mobile user that submits the request.

From the dummy profile, the identification of any dummy request that will be generated is known. In lines 7- 18, it is guaranteed that the total number of required dummies is less than *profileCount* and the proposed work generates all the dummy identifications from the dummy profile (line 12). Likewise, the temporal and spatial properties of the dummies are related to the mobile user (line 9, 10, 11).

```
/* Algorithm 1: Realistic-Dummy-Generation*/
1. precondition: mobileUserId!=null, totalDummies>0
2. input: mobileUserId, totalDummies,C /* C is query*/
3. method:
4. profileCount ←50, count ← 0 ,$x_{offset}$ , $y_{offset}$ , $t_{offset}$, dummies []
5./ * $x_{offset}$ , $y_{offset}$ , $t_{offset}$ are computed as shown in Section 8.6.1 */
6. profile [] = read_dummy_profile(mobileUserId,totalDummies)
7. if(totalDummies <= profileCount)
8.   while(count<totalDummies)
9.     t = C.t  + <0,1>* $t_{offset}$
10.    x = C.x + <0,1> * $x_{offset}$
11.    y = C.y + <0,1> * $y_{offset}$
12.    id = profile [count]
13     C' = diversify(C) /* diversifying the dummy query */
14.    newDummy = createDummy(id,x,y,t,C')
15.    dummies [count] = newDummy
16.    count++
17.    end while
18. end if
19. else
20. return dummies
21. end else
end
```

In line 13, to prevent the *homogeneity attack,* the query is *diversified*. For example, if the mobile request is related to 1036 N. Michigan Ave in Chicago, the dummy request may be related to 1038 N. Michigan Ave Chicago. The proposed work ensures that the dummies' query point is not the same as the real mobile users' query point. Instead, a different building or symbolic address in the region is utilized. The algorithm then creates the new dummy request based on the time, location, and query of the mobile user (line 14). In line 15, the new dummy is added to the list of dummies. Finally, the candidate list of dummies is returned in line 20.

In this algorithm, the dummy request becomes realistic and diverse by relating its temporal, spatial, and query point properties to the real mobile user's request.

8.6.2 Dummy diversity

If all the *K* users in the region submit request to the same symbolic address such as the same movie theater, K-anonymity fails, because the adversary can infer that some people are interested in that symbolic address. This kind of attack is referred to as the *homogeneity attack* [26]. For this reason the concept of diversity is considered [95]. Diversity adds another dimension to the privacy level and ensures that our query locations and query contents are diversified. For example, request should span across different postal addresses or different buildings. The dummy requests that are generated are diverse and are submitted to different symbolic addresses using a reverse geocoding scheme.

Using reverse geocoding, the location point of interests of the user is converted to a readable street address. Reverse geocoding is the process of converting a location point to an address or place name. For example, reverse geocoding latitude: 41.976216 and longitude: -87.90331, produces the address 99 Access Road Chicago, Illinois, 60666, USA. Based on the address or place that is returned, the proposed work constructs the dummies' point of interest. For example, a dummy point of interest for the aforementioned latitude and longitude would be at 100 Access Road, Chicago, Illinois, 606666, USA.

8.7 Query privacy in MobiPriv implies location privacy

We now show that anonymization with *K*-1 other mobile requests as done in *MobiPriv* is a sufficient condition for location privacy. First, let's define a query $q_m(x,y)(X,Y)$, where q is the query submitted by mobile user *m*, located at location (x, y), and the location component of the POI in the query is (X,Y). Both x and X corresponds to the latitude, y and Y corresponds to the longitude. We have two types of queries (1) **Type 1 -** The mobile user location is a component in the query. Therefore, (x, y) = (X, Y). (2) **Type 2-** The mobile user location is not a component in the query. Therefore, (x, y) != (X, Y). Location privacy ensures that (1) The location (x, y) is not revealed or (2) The location (x, y) is revealed and is indistinguishable from other mobile user location.

- Consider **type 1** queries, the (x, y) location of the mobile user is revealed in the query. In this case, the current mobile user location is a component of the query. For example, the query "*return the closest busses to my current location?*" In this case, the mobile user's current location constitutes the query. Therefore, $(x, y) \equiv (X, Y)$. In MobiPriv, we anonymize with regards to the query location point, in this case location (X,Y) is expanded to find *K-1* other mobile requests. Since $(x, y) \equiv (X, Y)$ and a mobile user cannot submit multiple requests simultaneously, the location (x, y) also become anonymous since the adversary only knows the location of some of the mobile clients in MobiPriv. For example, in Figure 50 (a), the entire grid represents the LBS world as seen by the anonymization server. The mobile users are M1, M2 and M3. They submit queries Q1, Q2 and Q3 respectively. The (row, column) pair represents the location of a mobile user or a location component of the query. Therefore, the locations of M1, M2 and M3 are (4, 0), (4, 1) and (0, 0) respectively. Likewise, the location component of the queries Q1, Q2 and Q3 are (4, 0), (2, 2) and (2, 3) respectively. Observe, only Q1 is a type 1 query, the remainder of this discussion is based on Q1. Mobile user M1 submits the query Q1 to the AS, with K=2, we perform spatial expansion of the query point to find other queries. The closest query is Q2; Q1 is therefore cloaked with Q2. We therefore submit a region consisting of Q1 and Q2 to the LBS. The query privacy achieved is 1/K (1/2) as required. The location of mobile client M1 cannot be deciphered by the adversary since we submit a region and the adversary only knows the location of some mobile users in the system.

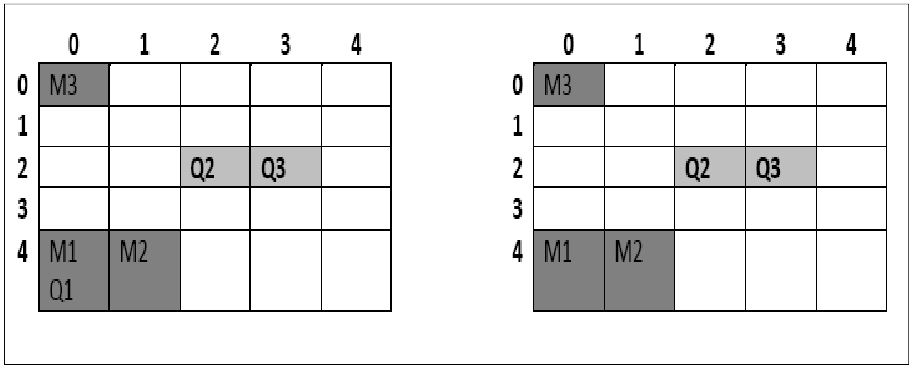


Figure 51 (a,b) -MobiPriv Query Privacy implies Location Privacy

8.7.1 Proof that query privacy implies location privacy for type 1 queries

For type 1 queries, the client's location M is a part of the query N. The proof is by contradiction, we assume to the contrary that protecting the query N would not protect the client's location M. Now, to protect N, the propose approach expand all the location components in N until K-1 other requests are found. Thus, the adversaries will observe that there are K interests in some location and has a 1/K chance of correlating the interests. Now, since we expand the query location points and M (i.e. the client’s location) is a component of the query, then M too would have been expanded and protected up to 1/K. This contradicts our initial assumption that protecting N would not protect M.

- Consider **type 2** queries, where the location of the mobile user is not a component of the query. For example, a mobile user submits a request for the closest parking lot to Point B, and the mobile user is located at Point A. In this case, there is no need for the mobile user to send his current location (Point A) to our anonymization server; hence location privacy is not breached. For example, in Figure 50 (b), Mobile user M1 submits a query Q2. Q2’s location point (2, 2) is not related to M1’s location (4, 0). For query privacy, we expand the region around Q2 to find Q3 and submit a region containing Q2 and Q3. Since M1’s location is not relevant to the query, M1’s location can be kept hidden.

This implies that regardless if the mobile user reveals their location in the query (i.e. type 1 queries) or not (i.e. type 2 queries), query privacy in MobiPriv always imply location privacy.

8.8 MobiPriv: System overview

MobiPriv is a three tier model similar to the trusted third party mechanism discussed earlier. The proposed suite of algorithms runs on the anonymization server. The first step is a *request submission* phase where the user submits a request. The request contains the percentage privacy level required by the user. Next, is the *transformation* phase where this personalized user percentage of privacy is converted to some value of *K* by a *mapping function*. Other phases include the *perturbation* phase whereby we form a region based on the spatial resolution from the request. The real user is then placed in the region. Next, we have two anonymization options, *CloakLessK* and *CloakedK*.

In *CloakLessK,* once the AS receives a mobile user's request it (i.e. the AS) quickly generates K-1 dummies in the perturbed region, and then sends the request to the LBS. In *CloakedK*, once a mobile user's request is received, before dummies are generated, the algorithm first verifies if other requests are close enough to this mobile user's request and can be cloaked together. Finally, if after cloaking with neighbor requests, *K-1* real mobile user requests are still not in the region, the proposed solution will generate realistic dummies as the remaining requests. Finally, the region request is sent to the LBS. The proposed algorithms maintain the following good contributions.

- All queries are given a response. In previous privacy models, some queries may not be given a response because *K-1* queries are not available to meet the QoS required, or *K-1* users are not available in the system. In these models [22, 60, 96,97], the mobile user defines their own personalized spatial tolerance, temporal tolerance, and privacy requirement. If the query requirement demanded by the mobile clients cannot be satisfied, the queries are dropped from the system. In MobiPriv, we have a drop rate of 0%. In the evaluation by experiments, s*uccess rate* is used as a matrix to measure reliability.
- Elimination of the temporal cloaking problem present in [9, 22, 60], whereby the system waits for K-1 other requests to be available before anonymizing the request. Some LBS systems provide emergency services and cannot tolerate delays.
- Elimination of the spatial cloaking problem whereby the system continues to extend the region to find K-1 other queries. The maximum size of the region is defined by the mobile user (i.e. *dx* and *dy*) from the query submission phase. We cannot expand the region beyond this restriction.

- Communication Cost Reduction, the proposed work presents two privacy aware models *CloakLessK* and *CloakedK*. *CloakLessK* has a fixed communication cost regardless of the number of mobile users submitting request. *CloakedK* has a much lower communication cost, the communication cost of *CloakedK* improves much more than *CloakLessK* as the number of request increases.
- Query linking in continuous queries is eliminated. The algorithms generate realistic and diverse dummy requests using a dynamic dummy profile strategy. This strategy eliminates this kind of attack. Experimental results revealed that the proposed work is resilient to this category of attack.

8.8.1 Mobile request

In *MobiPriv*, a mobile user submits a *request* in the form *<user_id, msg_num, {t, x, y}, {dx, dy, dt} ,P, C>*. The *user_id* is the unique identification of the userand *msg_num* to be the message identification. The combination of *user_id* and *msg_id* is unique for all messages. Also, *{t, x, y}* is the temporal and spatial property of the request. Additionally *dx, dy , dt* are the spatial and temporal resolution demanded by the mobile user. *P* is the percentage of privacy desired, and *C* is the request or query content. Recall, large spatial tolerances produces less accurate responses and high temporal tolerances result in longer message delays.

The point request above is converted to a region request of the form *<user_id', msg_num', {x1,x2}, {y1,y2}, {t1,t2}, C>* where *user_id', msg_num'* are hashed versions of *user_id*, and *msg_num* respectively. The parameters *x1, x2* are the request region’s *x* coordinates, *y1, y2* are the request region’s *y* coordinates, *t1* and *t2* represents the request region *z* coordinates. The three coordinates are used to form the cloaking box. *C* is the request content that should always be preserved. The size of the cloaking box is bounded as follows:

$x2 - x, x - x1 <= dx$

$y2 - y, y - y1 <= dy$

$t2 - t1 <= dt$

We refer to this process of converting a request from an exact point to a rectangular region as spatial and temporal cloaking.

8.8.1.1 Mobile request example:

<user_101, msg_num_004, {11:15am,-87.653, 41.85}, {60m, 50m, 5s}, 90% Privacy, "Shortest route from current location (latitude = -87.653, longitude = 41.85) to XXX rehab center (latitude = -87.6215, longitude = 41.210)">

8.8.2 Transformation and mapping function

MobiPriv allows mobile users to define the percentage value of privacy that they desire. Then it uses a mapping function to determine a suitable value of *K* based on the desired percentage privacy specified. The conversion from a percentage level of privacy to a suitable value of K is referred to as the *transformation* phase. AS administrators are not limited to one mapping function, they may define their own mapping function. A mapping function may reflect the nature of the underlying location based system. For example, in the proposed algorithm, a mapping function such that percentage privacy is related to the certainty with which an association between a user and a message can be ascertained is considered. Let *P* be the percentage of privacy desired by a real user U_r, a corresponding K-anonymity value is computed as follows:

$$K = ceiling\ (100/\ (100-P)),\ P < 100$$

8.9 Algorithms

We presented the realistic dummy generation algorithm in Section 8.6.1. Next, two other algorithms in MobiPriv, *CloakLessK* (i.e. Algorithm 2) and *CloakedK* (i.e. Algorithm 3) are discussed.

8.9.1 CloakLessK

First, the mobile user submits a request to the anonymization server (line 2). In line 4, we hash the user identification and the message identification. Each request contains the personalized privacy requirement of the mobile user that is converted to some *K*. This step is referred to as the transformation phase (line 5). The proposed algorithm then creates regions depending on *dx*, *dy*, *dt* (spatial and temporal tolerance) of the user request. The total number of users (real user and dummy users) in the region is determined by *K* from the *transformation* phase.

The real user is randomly placed in the bounding region (line 7), the AS then generates *K-1* dummies (line 8). The dummies generated are realistic and diverse. Realistic diverse dummies are used to protect against the corollary history attack. However, this protection comes with a cost (see experimental results). Finally, the cloaked region request is sent to the LBS for processing in line 9.

```
/* Algorithm 2: CloakLessK */
1. pre-condition: request!null
2. input: request <u_id,msg_num {t,x,y},{dx,dy,dt},P,C>
3. method:
4. hash (u_id ,msg_num )
5. K = transformation (P)
6. createGrid(request.dx, request.dy ,request.dt)
7. insertRealUser()
8. insertDummies (request.u_id, request.K-1)
9. sendRegionRequestToLBS()
10. end
```

Next, *CloakedK*, an algorithm that reduces the communication cost of CloakLessK is discussed.

8.9.2 CloakedK

The principal difference between *CloakLessK* (Algorithm 2) and *CloakedK* (Algorithm 3) is the addition of line 8 in the algorithm shown below. The input to the algorithm is a mobile user's request in line 2. In line 4, we hashed the identification parameters such as *user_id* and *msg_num*. The percentage of privacy demanded by the mobile client from line 2 is transformed to a suitable value of *K* in line 5. We create the bounding rectangle (region) around the location point in the request in line 6 and line 7.

Additionally, at line 8, of the *CloakedK* algorithm, instead of sending one real mobile user request along with dummies in a region request, the AS now aggregates multiple real user's requests in the same cloaking region before sending to the LBS. Only if real mobile users are less than K-1, then dummies are considered.

We now discuss the cloaking methodology of line 8. Once a user submits a request to the LBS, we generate a perturbed box as discussed in Section 8.6.1. The size of the box is dependent on *dx, dy, dt*. Before inserting any dummies inside the box, we query if this mobile query can be aggregated with other mobile queries currently in the system. Two mobile queries can be aggregated only if they are intersectable. If they can be aggregated, we then take a constraint on box size for the two queries. In line 9, the system verifies if the number of real mobile user request (*real_user_request_cnt*) in the cloaked region is less than the privacy requirement (K) of the mobile user that submitted the request. If *real_user_request_cnt* is less than K, the proposed algorithm generates the remaining request as realistic diverse dummy request. Finally, in line 12, the request is sent to the LBS for processing. The LBS processes the request and send the response to the AS, the AS then filters the response and sends the results to the client.

/ *Algorithm 3: CloakedK */

```
1. pre-condition: request!null
2. input: request <u_id,msg_num {t,x,y},{dx, dy, dt},P,C>
3. method:
4. hash (u_id ,msg_num )
5. K = transformation(P)
6. createGrid(request.K, request.dx, request.dy, request.dt)
7. insertRealUser()
8. intersectAndMerge()
9.  if (real_user_request_cnt < K)
10.   insertDummies()
11. endif
12. sendRegionRequestToLBS()
13. end
```

8.10 Complexity analysis discussion

In this section, the asymptotic complexity of the algorithms is discussed. Let *n* be the total number of requests in the system and K be the privacy requirement of a single mobile user submitting a new location based request.

In the case of CloakedK, the time complexity is bounded by the number of requests in the system. In line 8 of CloakedK, a linear search is performed to find at least K-1 other requests that are compatible with this incoming request. After this linear search if K-1 other requests cannot be found, then the outstanding requests are generated as dummy requests. Hence, the worst case complexity of CloakedK is O (n+(β *ε)), where ε is the time to generate a single dummy requests, and β is the number of outstanding requests to be generated.

In the case of CloaklessK, a search for other requests is not performed. Instead, the CloaklessK algorithm quickly generates K-1 other dummy requests instantaneously. Therefore, the complexity of CloaklessK is bounded by the time that it takes to generate the K-1 dummy requests (i.e. O((K-1) ε)). Consequently, CloaklessK should have a much better run time performance than CloakedK, especially for small K, and for large values *n*.

8.11 Proof of correctness for MobiPriv in snapshot queries

We will show that MobiPriv can satisfy any privacy requirement. By induction on the size of K, we will show that the proposed algorithm stays with the optimal solution, hence can guarantee high levels of location based privacy.

Let S = {r_2, r_3, r_4,… r_{k-1}, r_k} be the set of requests that are chosen to be anonymized with r_1 by the proposed algorithm for a privacy requirement K. Also, let T = {$r_2^{'}$, $r_3^{'}$, $r_4^{'}$…$r_{k-1}^{'}$, $r_k^{'}$} be the set of requests chosen by the optimal solution to be anonymized with r_1. That is, for the request r_1, the optimal solution chooses the requests $r_2^{'}$, $r_3^{'}$, $r_4^{'}$…$r_{k-1}^{'}$ and $r_k^{'}$.

For the case of K=1, the mobile client does not wish for their request be anonymized with other requests. In this case, both the optimal solution and the proposed solution would submit request r_1 only to the location based server. Assume true for the case of $K \geq 1$, that the proposed algorithm can satisfy a privacy requirement up to and including K (*inductive hypothesis*). Thus, for a privacy requirement of K, the hypothesis assumes that {r_1, r_2, r_3, r_4, … r_{k-1}, r_k} is satisfied for a snapshot. Now, for a privacy requirement demand of K+1, the optimal chooses a new request $r_{k+1}^{'}$ to be added to T. As a result, if we added $r_{k+1}^{'}$ to the set S (i.e. {r_1, r_2, r_3, r_4…r_{k-1}, r_k}) that was chosen by the proposed work according to the hypothesis, the privacy requirement of K+1 is satisfied correctly. However, for the $(K+1)^{th}$ privacy demand, the proposed algorithm will generate a new dummy request r_{k+1} to satisfy the privacy requirement. Therefore, the inclusion of r_{k+1} to $\{r_1, r_2, r_3, r_4,\ldots r_{k-1}, r_k\}$ (i.e. *inductive hypothesis*) will satisfy the privacy requirement of K+1.

8.12 Evaluation

We performed experimental evaluations of the proposed algorithms (*CloakLessK* and *CloakedK*) against the three PrivacyGrid approaches (*Bottom Up, Top Down, Hybrid*) [22] and also against the *pyramid* based approach such as Casper [9]. First, the three PrivacyGrid techniques [22] are briefly discussed. In the *bottom-up* cloaking the mobile user cell is expanded to meet the K-anonymity and diversity requirement. Mobile users are considered for the K-anonymity count and static objects e.g. gas stations, supermarkets are used for the diversity count. For a given cell it may expand to its immediate neighbor east, west, north, or south. The next cell to be chosen is the cell with the highest mobile user count. This cell will be included in cloaking box. The *top-down* approach first selects the largest possible cloaking box that satisfies the users QoS requirement. If *K* users cannot be found in this cloaking box the query cannot be satisfied. If *K* mobile users are found, the algorithm will attempt to prune the cloaking box to see if *K* can still be satisfied. If K can still be satisfied in a smaller cloaking box then the latter and smaller cloaking box is chosen. The *hybrid* approach makes a decision to use the bottom up, or top down, depending on the value of *K* and the QoS. Hybrid combines the strength of both (top down and bottom up) approaches.

The pyramid based scheme used [9] is now discussed. In [9], for anonymization and cloaking a pyramid structure is maintained. The cells in the region maintain the number of mobile users present in each cell. If the current cell of user cannot satisfy the value of *K*, then neighboring cells are considered. A cell is considered a neighbor if they have the same parent. If expansion to neighbors does not satisfy the user requirements, then the parent expansion is considered. One can envision a balance quad-tree where the users are leaves in the system.

8.12.1 Evaluation criteria

In this section, the evaluation criteria that we used to evaluate the efficiency and effectiveness of *MobiPriv* and previously proposed algorithms are discussed.

8.12.1.1 Success rate

One of the most important evaluation criteria is the *success rate*. The main goal of any anonymization server is to maximize the number of messages that can be successfully anonymized with the personalize quality of service and privacy requirement desired. The *success rate* is measured as the ratio of the number of successful anonymized request, by the total number of incoming mobile request.

A success rate of 100% implies that all the requests that are sent by the mobile clients are safely anonymized. In some systems [9, 22, 60, 19, 66], the request is dropped because the privacy requirement cannot be satisfied. Let *N* be the total number of mobile requests send to any cloaking algorithm *CloakM*. Then, the set of mobile requests that can be successfully anonymized with the personalized quality of service and privacy requirement can be calculated as $\{m_t \mid m_t = CloakM\ (m_s), m_s \epsilon N\}$ where m_s is the request sent to $CloakM\ (m_s)$. The success rate of any algorithm $CloakM\ (m_s)$ is given by:

$$Success\ rate = \left(\frac{|\{m_t \mid m_t\ = CloakM\ (m_s\ \in N)\}|}{|N|}\right) *\ 100$$

8.12.1.2 Performance measure

The run time performance of the algorithms is measured as the cloaking time. The cloaking time of an algorithm is the time taken to perturb and privatize the mobile requests. An algorithm with a lower cloaking time does better, because the cloaking time is a measure of the temporal complexity. Efficient cloaking implies that the algorithm spends less time processing the incoming mobile requests from the mobile clients. We define a function *startTime (cloaking Algorithm)*, that returns the time the cloaking algorithm start the anonymization process. Also, *endTime (cloakingAlgorithm)*, which returns the time the cloaking algorithm completes the anonymization process.

$$\text{cloaking time} = endTime(CloakM(m_s)) - startTime(CloakM(m_s))$$

8.12.1.3 Communication cost

We were interested in measuring the communication cost of the proposed algorithm and previously proposed algorithms. The communication cost is a measure of the number of messages sent by the anonymization server to the LBS. For example, in CloakLessK, for each request received from the mobile client, the algorithm immediately aggregates that request with dummy requests and then forwards the aggregated request (i.e. region request) to the LBS. Therefore, if N requests are submitted to the anonymization server, then we also have N region requests being submitted to the LBS. In the case of other cloaking algorithms such as Casper [9], CloakedK, or PrivacyGrid [22], for N requests sent to the anonymization server, the anonymization server may send less than N region requests to the LBS. In these schemes, multiple real requests can be aggregated together and forwarded to the LBS. Let N be the total number of mobile request submitted at time t by the mobile clients. The number of *region requests* submitted by the anonymization server (AS) for the N incoming request is *m*, *1<=m<=N*. We measure the communication cost as the ratio:

$$communication\ cost = \frac{m}{N}, 1 \leq m \leq N$$

8.12.1.4 Quality of service

Some QoS evaluation variables considered are *spatial tolerance* and *anonymity level*. Spatial tolerance is the user defined spatial resolution that should be satisfied in conjunction with the anonymity level. The anonymity level is the user defined K-anonymity requirement.

8.13 Experimental setup and road network

The experiments are conducted on a Windows machine running the P8400 Intel DUO 2.27 GHz processor with 4GB of RAM. The six algorithms (Bottom up PrivacyGrid, Top down PrivacyGrid, Hybrid PrivacyGrid, Casper Pyramid approach, MobiPriv CloakLessK, MobiPriv CloakedK) were implemented using Java. We refer to these algorithms as *B*, *T*, *H*, *Py*, *CLK*, and *CK* respectively in the experiments.

The mobile object generator that was considered is an extension of mobile object generator used in [22, 60]. A map of Chamblee in the state of Georgia USA was used for the experiments. The map covers a region of 160 km^2, see Figure 51. Moving object traces were generated based on real world traffic volume data extracted from [59] for 10,000 cars traveling along the road network. Three types of roads are considered in the simulation; expressway, arterial, and collector roads (see Table 13). Cars are placed randomly on the road network initially, and then continue to move along a road trajectory making a decision at each intersection. The properties of each road type considered in the experiment are shown in Table 13. Each car (mobile user) generates multiple requests to the anonymization server (AS) running MobiPriv algorithms. The anonymization server then anonymizes the request before forwarding it to the service provider (LBS).

Table 13- Road Properties

Properties	Road categories		
	Expressway	*Arterial*	*Collector*
Mean speed (km/h)	90	60	50
Std. Dev (km/h)	20	15	10
Traffic volume (cars/h)	2916.6	916.6	250

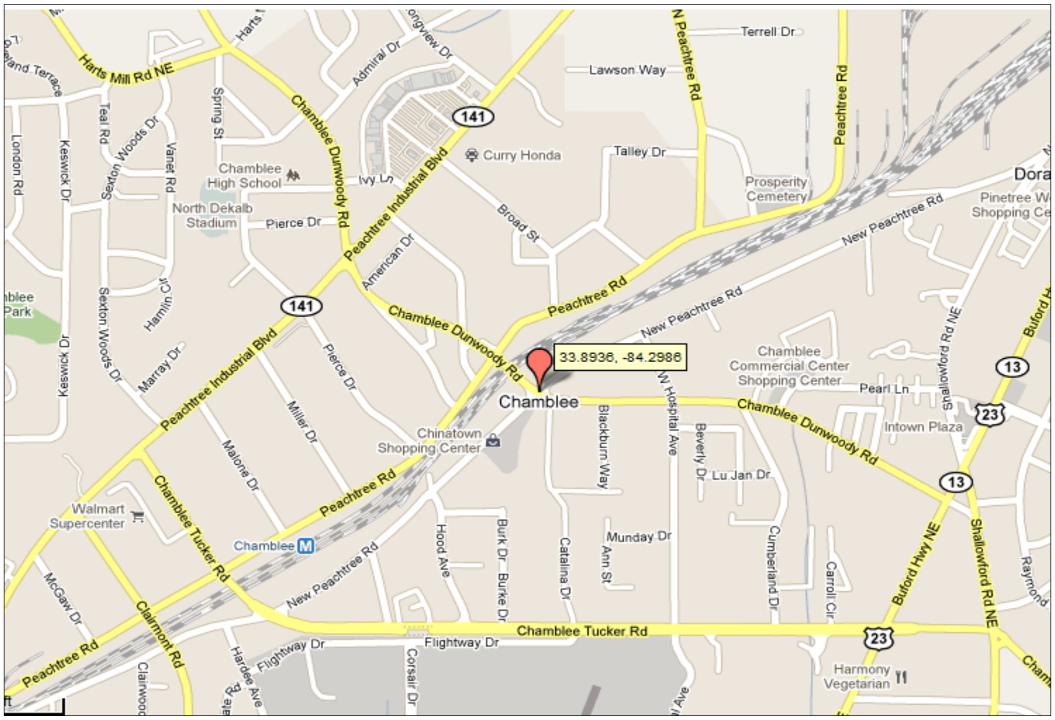


Figure 52 - Evaluation area (Map of Chamblee Region, Georgia, USA)

8.14 Experimental results

The experimental results for the proposed algorithms compared against previously proposed algorithms are presented in this section. The algorithms' effectiveness and efficiency under varying user requirements are studied.

8.14.1 High privacy requirement and high spatial tolerance

For these experiments, the algorithms' performance when the mobile user demands a high privacy level and a high spatial tolerance is studied. A high spatial tolerance implies a low quality of service. In Figure 52, 53, and 54 we evaluate the success rate, cloaking time, and communication cost with *high spatial tolerance* (i.e. 700m*700m) and *high anonymity level* (i.e. K=50).

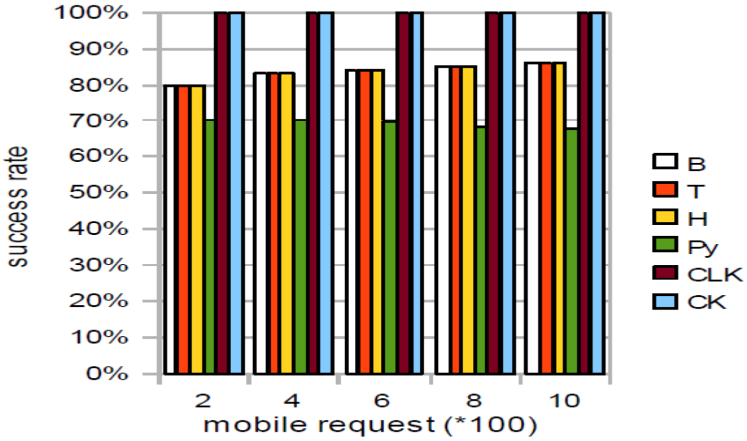


Figure 53-success rate and number of mobile requests

Figure 52 plots the success rate with varying number of mobile requests sent to the anonymization server. The x- axis represents the number of mobile request sent by the mobile object generator to our anonymization server. The y-axis represents the success rate of the algorithms in percentage. We saw that *MobiPriv* algorithms achieved 100% success rate. This implies that all the requests sent can be safely anonymized with the *MobiPriv* algorithms.

The PrivacyGrid (top down, bottom up, hybrid) [22] and Pyramid approaches [9] only anonymized 70%-80% of the mobile requests sent to the anonymization server. PrivacyGrid and Pyramid schemes will drop some of the requests because the QoS demanded by the client cannot be satisfied. An increase in the number of mobile requests sent to the anonymization server showed a slight increase in the success rate of the other algorithms (*B*, *T*, *H*, *Py*). This make sense, since there are more requests in the system, it is easier to locate K-1 other request to anonymize with even under tighter constraints.

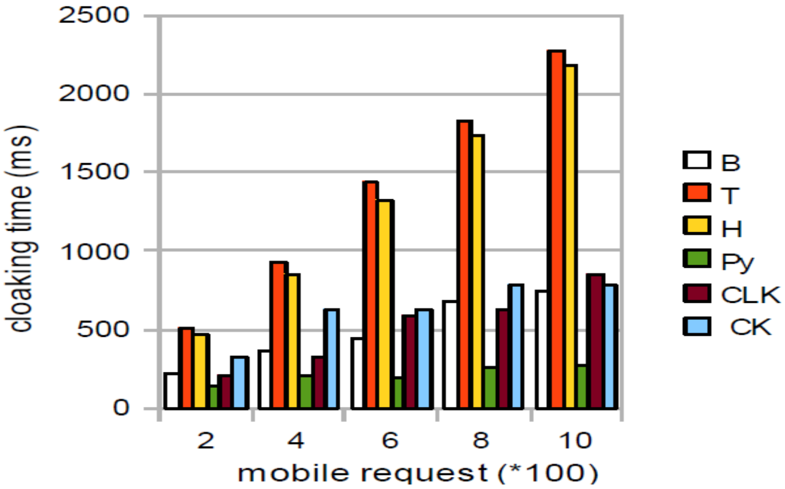


Figure 54- cloaking time and number of mobile requests

Figure 53 expresses the run time performance against increasing number of mobile requests sent to the anonymization server. We refer to the run time performance as “cloaking time”. *MobiPriv* algorithms achieve fast cloaking time. In particular, *CloakLessK* has a lower cloaking time than *CloakedK* since it can quickly generate K-1 dummy requests instead of searching for real requests initially. The fastest cloaking time under these settings is the pyramid based in [9].

Also, MobiPriv algorithm's cloaking time is hardly affected by an increase in the number of mobile requests. On the other hand, the cloaking time of the PrivacyGrid schemes are severely affected by an increase in number of mobile user request. As the number of requests increases from 200 to 1000, the cloaking time of the PrivacyGrid schemes [22] increases rapidly. This is not the case for the proposed MobiPriv algorithms.

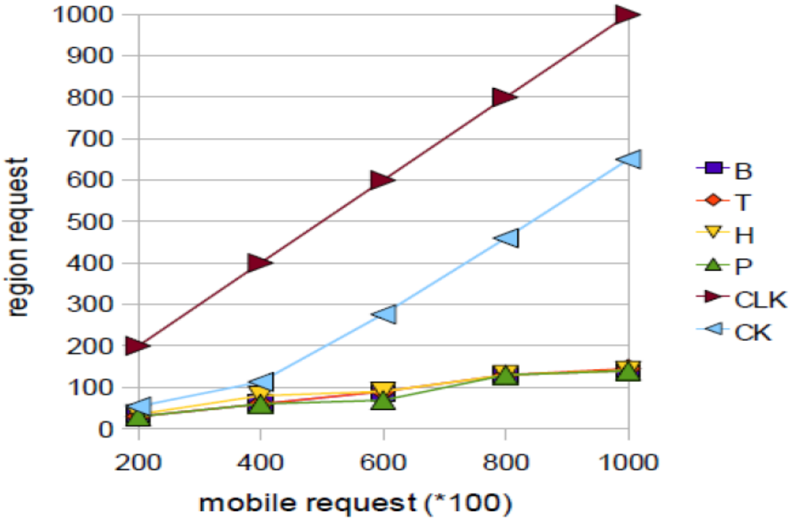


Figure 55-communication cost and number of

In Figure 54, the graph depicts the communication cost. We saw that MobiPriv algorithms have a higher communication cost. More specifically, CloakedK does better than CloakLessK and continues to do much better as the number of request increases.

We conclude the discussion of high anonymity level and high spatial tolerance by claiming that MobiPriv algorithms guarantee that all messages can be anonymized safely at a relatively fast speed for high anonymity level and high spatial tolerance. MobiPriv algorithms on the other hand have higher communication cost. *MobiPriv CloakedK* has a better communication cost than

CloakLessK. Additionally, the other trusted third party schemes such as PrivacyGrid [22] and Pyramid based [9] dropped 20%-30% of the user requests. In these schemes, if the privacy requirement and QoS cannot be satisfied, the request is dropped. MobiPriv algorithms on the other hand are a guarantee service, they will never drop a request. Instead, MobiPriv always achieves 100% success rate.

8.14.2 Lower privacy requirement and high spatial tolerance

For these experiments, the algorithms' performance when the mobile user demands a lower privacy level and a high spatial tolerance is studied. We evaluate algorithms' success rate, cloaking time, and communication cost with *low average anonymity level* (K=10) and *high spatial tolerance* (700m*700m) in Figures 55, 56, and 57. The primary difference between these experiments and the previous experiment (Figure 52, 53, and 54) is the fact that the privacy requirement is now *K=10* instead of *K=50*.

Figure 55 plots the success rate with different number of user requests. In general, all cloaking algorithms should have a higher success rate with a reduction in the average anonymity level (i.e. from K=50 to K=10). *MobiPriv* algorithms still anonymized all the requests. This is not the case for the other algorithms (B, T, H, Py). However, the three PrivacyGrid and the Pyramid based systems showed an improvement in the success rate. This improvement in success rate is because it is less challenging to discover a smaller number (*K=10*) of requests to cloak with than K=50. Pyramid scheme anonymizes 98.5% of the total request received, this indicates that a few of the requests are still discarded. MobiPriv algorithms achieve 100% success rate, it does not discard requests.

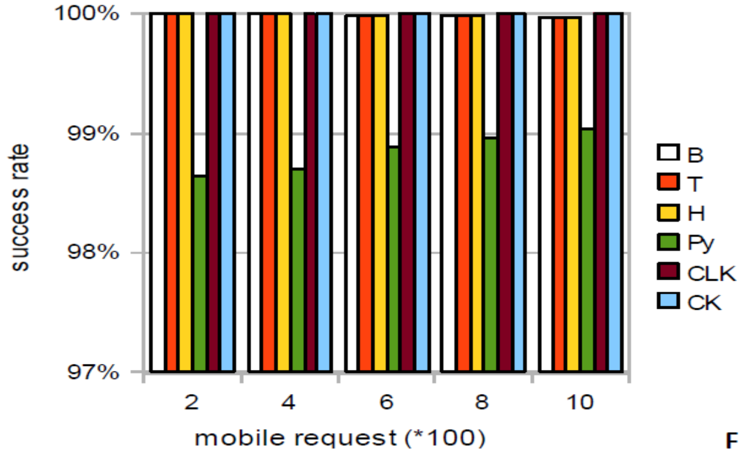


Figure 56 - success rate and number mobile

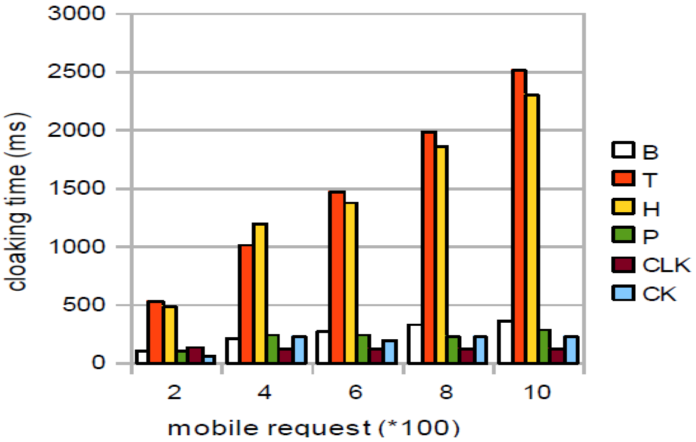


Figure 57 - cloaking time and number of mobile requests

In Figure 56, the graph shows the cloaking time (run time performance) on the y-axis with increasing number of mobile request on the x-axis. The proposed *MobiPriv* algorithms have good cloaking time. In particular, *CloakLessK* has an exceptionally fast cloaking time for low anonymity level (e.g. K=10) since it can quickly generate dummies. While the run time performance of the other schemes, such as PrivacyGrid's top down and hybrid approaches is very high.

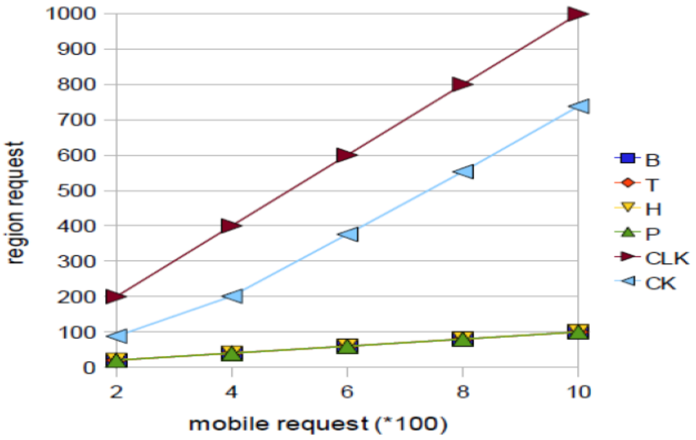


Figure 58 - communication cost and number mobile

From Figure 57, all the privacy algorithms submit more region requests to the LBS with an increase in the number of incoming requests. *CloakLessK* communication cost is not affected by a reduction in anonymity level. We end the discussion of the algorithms under low anonymity level and high spatial tolerance by concurring that MobiPriv algorithms still maintain a 100% success rate and also the fastest cloaking time. However, we pay a price for communication cost.

8.14.3 Lower privacy requirement and low spatial tolerance

For these experiments, the algorithms' (i.e. B, T, H, P, CLK, CK) performance when the mobile user demands a lower privacy level and a low spatial tolerance is studied. A low spatial tolerance implies a high quality of service. We evaluate the algorithms under *low average anonymity* level (i.e. K=10) and *low spatial tolerance* (100m * 100m) in Figures 58, 59, and 60. This experiment studies the algorithms when the mobile clients demand high quality of service.

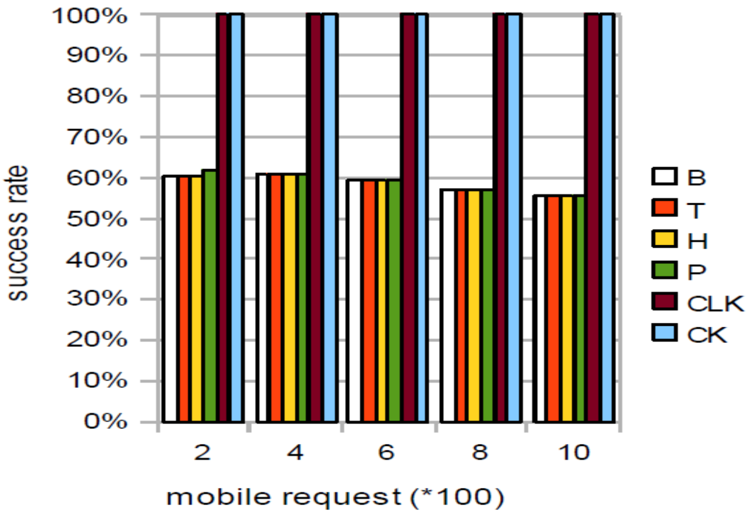


Figure 59 - success rate and number of mobile

Given a low spatial tolerance (i.e. 100m * 100m) and low anonymity level, Figure 58 shows the success rate with different number of request. We make two important observations. First, the success rate of the PrivacyGrid and Pyramid approaches is less than 60%. This implies that these algorithms running on the anonymization server dropped over 40% of the total mobile requests received. This is undesirable, especially in critical location based systems. Secondly, MobiPriv algorithms still maintains a 100% success rate even under such high personalized QoS requirements. This is one of the main strengths of MobiPriv, it never discard a request. The main reason why the previous approaches [22, 9] drops over 40% of the incoming location based requests is because the privacy (i.e. K) and the QoS (i.e. spatial tolerance) requirements cannot be satisfied.

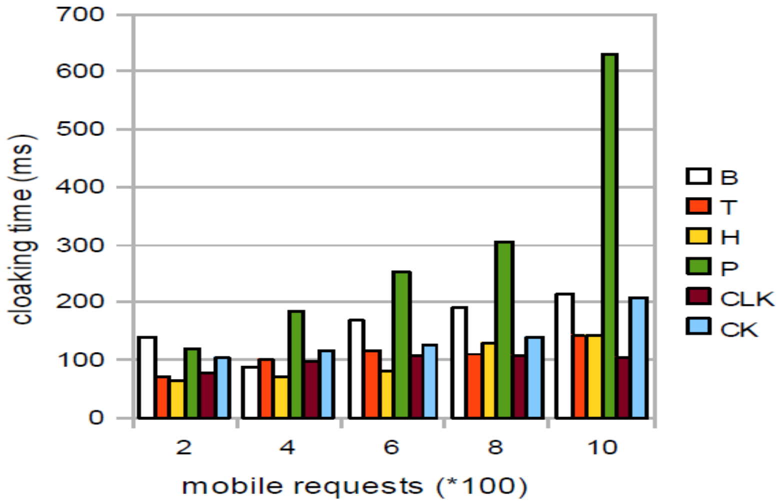


Figure 60-cloaking time and number of mobile

Figure 59 plots the run time performance (cloaking time) with varying number of mobile requests. The most obvious observation is that the pyramid scheme now has a very high cloaking time for low spatial resolutions. The *MobiPriv* algorithms have a fast cloaking time. Specifically, *CloakLessK* does slightly better than *CloakedK*. The runtime of the other algorithms (i.e. B, T, H, Py) are poor if we consider that they only anonymized 60% of the total incoming mobile requests.

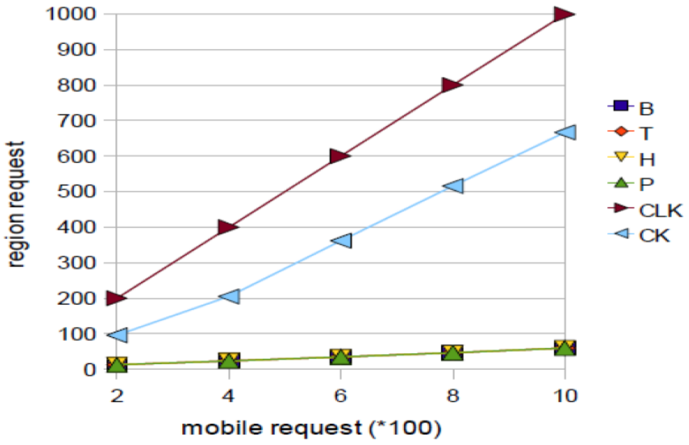


Figure 61 - communication and number of mobile requests

Figure 60 highlights the communication cost. It is observed that *CloakedK* has a better communication cost than C*loakLessK.* In general, *MobiPriv* algorithms have a higher communication cost, it ensures that all the messages are anonymized. The other algorithms (i.e. B, T, H, Py) anonymize 60% of the requests. Consequently, the other 40% was discarded, thus did not contribute to the communication cost.

We end the discussion of low average anonymity level and low spatial tolerance by claiming that MobiPriv algorithms maintain a 100% success rate while the PrivacyGrid and Pyramid based approaches achieved 60% success rate. Thus, MobiPriv guarantees a 100 % success rate under high personalized quality of service or privacy requirements. The proposed MobiPriv algorithms have a fast cloaking time that is comparable to any of the algorithms that we studied. Specifically, *MobiPriv CloakedK* does better in communication cost than *CloakLessK*. Additionally, the communication cost of the other schemes (i.e. B, T, H, Py) is better than MobiPriv’s.

8.14.4 Privacy and Quality of Service (spatial tolerance)

For these experiments, we study the algorithms' performance under variety of spatial tolerances. In Figures 61, 62, and 63, the effects of spatial resolution (QoS) on the success rate, performance (i.e cloaking time) and communication cost is evaluated. The average anonymity level is set to K=50, and the number of mobile requests submitted to the anonymization server is 1000. Figure 61 plots the success rate with the personalized spatial tolerance. Again, *MobiPriv* algorithms achieve 100% success rate for any spatial tolerance, even extremely low (i.e. less than 50m) spatial tolerance requirements. The *PrivacyGrid* and *Pyramid* approaches all have very low success rate for low spatial tolerance. For a spatial tolerance of 50m*50m, the *PrivacyGrid* (top down, bottom up, hybrid) [22] and pyramid based (Casper) [9] dropped over 98% of the total mobile requests sent to the anonymization server. With an increase in spatial resolution to 100m*100m, PrivacyGrid and Pyramid schemes still drop 70% of the total incoming location based requests.

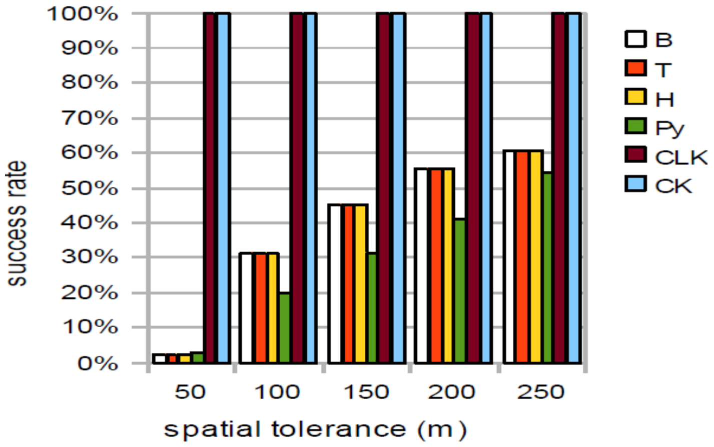


Figure 62 - success rate and quality of service (spatial tolerance)

Figure 62 shows the cloaking time with different spatial tolerances. The *MobiPriv* algorithms all have a higher cloaking time because all the mobile requests that are sent to the anonymization server are anonymized successfully, unlike the other algorithms (i.e. B, T, H, Py) that only anonymize a small fraction (i.e. 2%) of the total mobile requests.

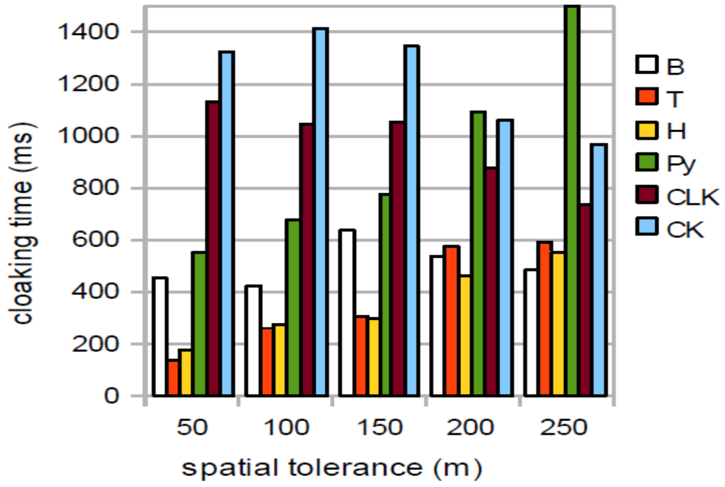


Figure 63 - cloaking time and spatial tolerance

Figure 63 highlights the chart of communication cost with different spatial tolerances. From Figure 63, it is obvious that the spatial tolerance does not affect the communication of *CloakLessK*. As the spatial tolerance increases, the communication cost of *CloakedK* decreases. This is related to the fact that more real mobile users can be included in the *region request.* The other schemes, PrivacyGrid and Pyramid, all have low communication cost because most of the requests that they received are discarded.

We conclude the study on the effects of low spatial tolerance on success rate, cloaking time and communication cost. The study showed that under low spatial tolerances (e.g. 50m*50m)

PrivacyGrid and Pyramid schemes discard most (98%) of the mobile request received. An example of a low spatial requirement in real life location based system is a transit itinerary system where the user of the system may request the shortest or fastest route from an origin to a destination and does not want to walk more than 50m. Additionally, apart from the high success rate, the MobiPriv algorithms have good run time performance.

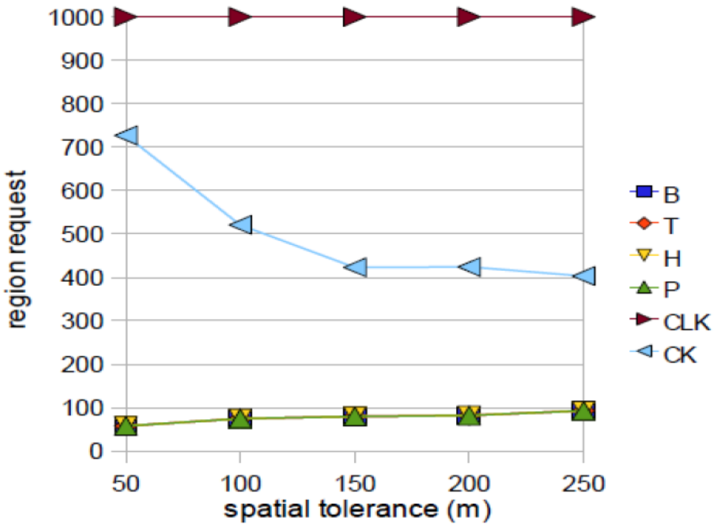


Figure 64 - communication cost and spatial tolerance

8.15 Continuous queries

In this section, continuous queries are studied. We will show that current snapshot solutions cannot overcome the privacy challenges in continuous querying systems. The proposed technique and experimental results for privacy preservation in continuous location based queries are presented.

8.15.1 Query linking in continuous queries

Some anonymization techniques that involve a trusted third party AS [22, 9, 60] cannot protect against the corollary history attack. In these models, when a user *u* submits a query to the AS there is a search for K-1 other requests. If the K-1 other requests are not found immediately, the cloaking region is expanded with the intention to locate these other requests. If *u* moves to another location and submits another query, *K-1* other requests have to be rediscovered again. However, the *K-1* requests in the latter region request may be different from the former *K-1* requests. Consequently, taking the intersection of the latter and the former regions, enables the query to be linked to the mobile client *u* in such systems. Furthermore, rediscovery of the same user requests for subsequent requests reduces the QoS as observed by Chow et al [66].

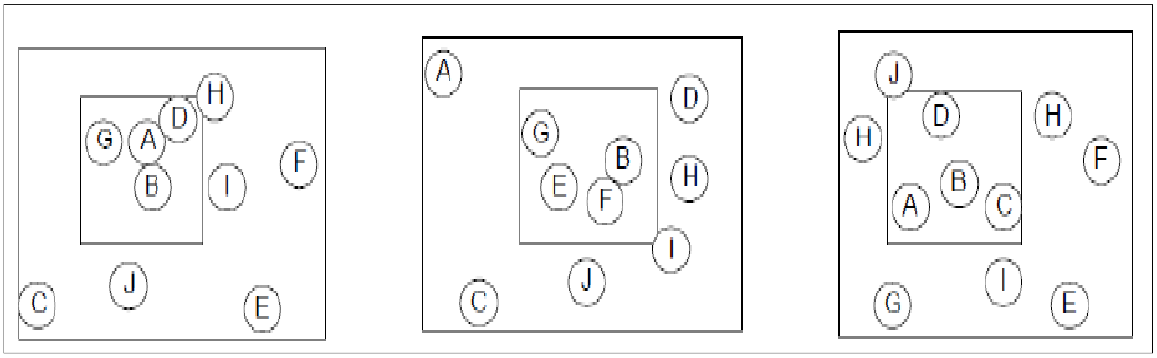


Figure 65 (a, b, c) - Corollary history attack in continuous queries

Consider the following scenario in Figure 64 with 10 mobile clients (i.e. *A, B, C, D, E, F, G, H, I, J*). The queries are submitted by mobile client B and consist of three discrete timestamp readings t0, t1, and t2 (i.e. Figure 64 (a), Figure 64 (b) and Figure 64 (c)) respectively. The large rectangles in Figures 64(a, b, c) represent the map of the "world" where all the mobile clients reside. The small rectangles represent a region request, it contains the mobile users that are anonymized together. The desired local anonymity level for mobile user B corresponds to K=4 for explanation purposes.

At time t0 (i.e. Figure 64a), the K-anonymity value of 4 submitted by the mobile client *B* is satisfied because mobile client *B* is anonymized with mobile clients *A*, *G*, and *D* as shown by the small rectangle in Figure 64 (a). Also, at time t_1, the local *K-anonymity* of the mobile client is satisfied (K =4). However, the adversary may take the intersection of the two snapshots (t0, t1) and conclude that only mobile clients *B* and *G* are present in both snapshots. Hence, the query linking is reduced to ½. Further, at time t_2, if the intersection of all three snapshots (t_0, t_1, t_2) is taken, mobile client *B* will be positively linked to the query. We refer to this query linking in continuous queries as the *corollary history attack*. Many snapshot solutions [9, 22, 60, 93] cannot overcome this category of attack since previous anonymize candidates are not taken into consideration by these models. Then we will show how MobiPriv prevents this kind of attack.

8.15.2 Discussion on K_{global} and K_{local}

For continuous queries users are expected to define two privacy requirement parameters, K_{global} and $K_{local.}$ K_{local} is the local privacy for each snapshot, and K_{global} is the global privacy across snapshots. In snapshot systems we refer to K_{local} as K.

Given K_{global} and region requests $R_1, R_2 \ldots R_{n-1}, R_n$ MobiPriv ensures that the following holds.

- $|\{R_1 \cap R_2 \cap \ldots \dot{R}_{n-1} \cap R_n\}| \geq \mathrm{K}_{\mathrm{global}}$
- $\mathrm{K}_{\mathrm{local}} \geq \mathrm{K}_{\mathrm{global}}\ \forall \mathrm{K}_{\mathrm{local}}$ where $\mathrm{K}_{\mathrm{local}}$ is the local snapshot anonymity

8.15.3 MobiPriv prevents corollary history attacks

Let *U* be the set of real users in the system. For each real user $U_r \in U$, we maintain a dummy profile. This dummy profile associates U_r with a set of *profileCount* amount of dummy users A_r. When a user submits the required privacy level as a part of the request, we associate this privacy level with a value of *K* in K-anonymity via a phase we refer to as transformation. If the derived of value K < *profileCount* we generate *K-1* dummies such that the *K-1* dummies $\subset A_{r.\ A}$n example is discussed below.

Assume a user U_r submits a query to the AS with a privacy requirement corresponding to K_{local}=5. Let the privacy profile (dummy profile) of U_r be the set {U_{a1} , U_{a2} , U_{a3} , U_{a4} , U_{a5} , U_{a6} , U_{a7} , U_{a8} } where *profileCount*=8. In the first region request R_1, we maintain the set {U_r , U_{a1} ,U_{a2} ,U_{a3} , U_{a4} }. Assume U_r submits another query in R_2, with a privacy requirement corresponding to K_{local} =6 we maintain the set {U_r, U_{a1} , U_{a2} ,U_{a3} ,U_{a4} , U_{a5}}. If U_r submits a third query in cloaking region 3 (R_3) with a privacy requirement corresponding to K_{local} =7, we maintain the set {U_r , U_{a1} ,U_{a2} ,U_{a3} ,U_{a4} , U_{a5},U_{a6} }.

Observe, if the adversary takes the intersection of $R_1 \cap R_2 \cap R_3$ = {U_r , U_{a1} ,U_{a2} ,U_{a3} ,U_{a4} }. This implies that the user has a 1/5 chance of been identified, regardless if the adversaries can aggregate region requests across multiple regions. Clearly, there is a correspondence between the lowest value of K specified by the user and the chance of been linked to a query in *MobiPriv*.

In general, for region requests R_1, R_2, R_{n-1}, R_n with corresponding privacy requirements K^1_{local}, K^2_{local}, K^{n-1}_{local}, and K^n_{local} the below constraint holds.

$$|\{R_1 \cap\ R_2 \cap \ldots R_{n-1}\ \cap\ R_n\}|\ \leq min\ (K^1_{local}\ , K^2_{local}, \ldots, K^{n-1}_{local}, K^n_{local})$$

We prove this by contradiction. Since K_{local} determines the number of elements in R, the intersections cannot contain more than the number of elements in any set. Let $\{R_1 \cap\ R_2 \cap \ldots R_{n-1}\ \cap\ R_n\}$ be the set of regions and $K_{min} =\ min_{1\leq i\leq n}\ \{|R_i|\}$. Let A = $\{R_1 \cap\ R_2 \cap \ldots R_{n-1}\ \cap\ R_n\}$. Suppose to the contrary $|\mathrm{A}| > K_{min}$, we know that all elements of A are contained in all sets R_i $1 \leq i \leq n$ by the definition of intersection. Therefore, all sets R_i $1 \leq i \leq n$ have at least |*A*| elements, which contradicts $|\mathrm{A}| > K_{min}$

8.15.4 Proof of correctness for continuous queries

We will prove by induction on the size of n, where n is the number of region requests, that for any number of requests, MobiPriv can satisfy the global privacy constraints. In previous sections of the dissertation, we proved that MobiPriv can satisfy the local constraints using induction. For any value of n, the proposed algorithm can guarantee both local and global privacy. Therefore, for n = $|\{R_1 \cap\ R_2 \cap \ldots R_{n-1}\ \cap\ R_n\ \}|$ we will show that MobiPriv will preserve global privacy across all the snapshots and also show MobiPriv algorithm is aligned with the optimal solution.

Let S = $\{R_1, R_2, \ldots R_{n-1}\ R_n\}$ denote the set of regions generated by the proposed algorithm, and T = $\{R_1{}', R_2{}', \ldots R_{n-1}{}'\ R_n{}'\}$ be the set generated by the optimal solution. This is true for the case of n=1. For this case, we only have one region request (i.e. R_1) been submitted by user u_{real}. The optimal solution would have anonymized with any K_{local}-1 amount of different requests if they are available. Likewise, in MobiPriv, we generate at least K_{local}-1 different dummy requests for the first snapshot (i.e. R_1). Thus, K_{local} for the first snapshot is satisfied. For both the optimal and the proposed algorithm, the global privacy is not taken into account since *n=1*.

Assume that it is true for some value $n \geq 1$(*inductive hypothesis*). This implies that the algorithm's first *n* regions have properties of the optimal. That is, the local K-anonymity is satisfied and the global anonymity is satisfied up to and including R_n.

As a result, if we added $R_{n+1}{}'$ *(i.e. n+1th optimal region)* to T $\{R_1, R_2, \ldots R_{n-1}\ R_n\}$ (i.e. *inductive hypothesis*) we would not violate the local or global constraints. But in the (n+1)th region, the proposed algorithm generate the same set of dummies as in previous regions (i.e.

$\{R_1, R_2, \ldots R_{n-1}\ R_n\}$) for the (n+1)th region. Hence, taking the intersection of hypothesis $\{R_1, R_2, \ldots R_{n-1}\ R_n\}$ and R_{n+1} will not violate the local or global constraints.

8.15.5 Resilience in continuous queries

In this section, MobiPriv's privacy preservation guarantee for continuous queries is studied.

8.15.5.1 Resilience

This evaluation matrix is a *percentage* measure of the resilience to the corollary history attack that causes query linking in continuous queries. As we explained previously, corollary history attack is present if multiple mobile requests are submitted by the same mobile user. Consider, a mobile user submitting requests U with a privacy level of *K*=5. If for the first snapshot, we get (U, B, C, D, E), where B, C, D, and E are other requests that request U is anonymized with. For the 2nd snapshot we get (U, B, C, G, H). Globally, we have an overlap of three common mobile requests, U, B, and C. Therefore, the resilience to the corollary history attack in this case becomes 3/5 or 60%. Formalized as follows; let the personalized privacy requirement be *K* in K-anonymity. Also, let the number of common clients in the different snapshots be given by the function *intersect (SN1, SN2, SN3…SNn)*, where *SN1* is the first snapshot, and *SNn* is the last snapshot submitted by the same mobile client.

resilience = (|intersect (SN1, SN2, SN3…SNn)|/ K) * 100

In this section, the algorithms' resistivity to the corollary history attack in continuous queries is evaluated. As mentioned previously, some anonymization techniques cannot protect effectively against corollary attacks because these algorithms greedily selects the nearest mobile requests, without any regard for previous anonymized regions. Mobile requests from previous anonymized regions should be used as selected candidates for future anonymized region requests.

8.15.6 Resilience across snapshots

Figure 65 highlights the graph of resilience against the number of queries. The number of mobile requests in the system is 10000, the average K value in K-anonymity is 5 and the average spatial tolerance is 100m * 100m. Recall, for corollary history attacks to be possible, a single mobile user must submit multiple requests at discrete time points to our anonymization server. In Figure 65, the mobile user issues 5 distinct location based requests.

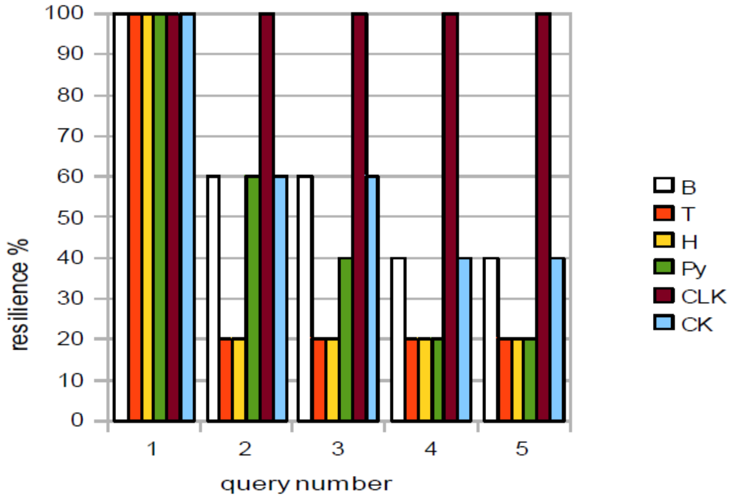


Figure 66 - resilience across five snapshot queries within a continuous query

We measure the resilience of each request using previously anonymized requests as candidates for the current request. From Figure 65, it is observed that for the first query issued, the resilience of all the algorithms is 100% because there are no previously anonymized entrants. In general, CLK algorithm has the best resilience because it will always generate the same set of dummies for mobile requests from the same mobile user. Also, for the other algorithms, the resilience decreases as the number of queries increases since it becomes more challenging for the intersection of the different region requests to contain the same mobile requests.

8.15.7 Resilience and Quality of Service (spatial tolerance)

Figure 66 depicts the graph of average resilience against spatial tolerance. The mobile client submits 5 mobile requests similar to Figure 65, and we then take the average resilience of the 5 mobile requests under varying spatial tolerance. The number of mobile requests in the system is 5000, the average K value =5, and we vary the average spatial resolution from 50m*50m to 500m * 500m. First, we observe that for high quality of service (low spatial tolerance) such as 50m * 50m, the resilience of *B*, *T*, *H* and *Py* is 0 because the privacy requirement (K=5) and QoS (50m * 50m) desired cannot be satisfied, hence, the query is discarded.

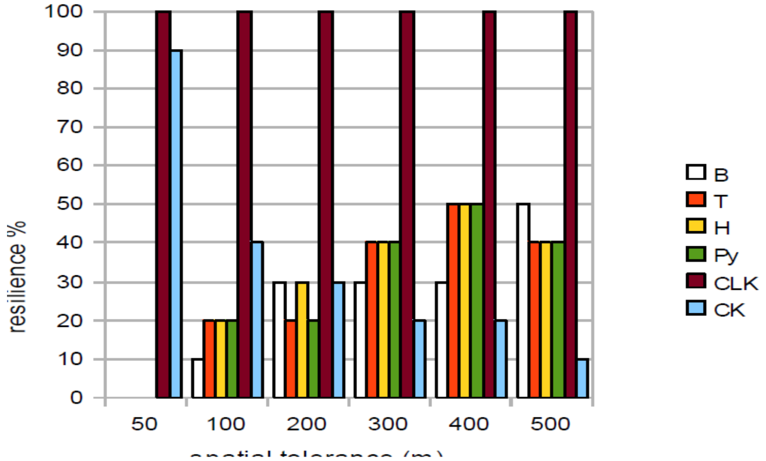


Figure 67 - resilience and quality of service

More specifically, *CLK* algorithm has the best resilience for any QoS desired. The *CLK* algorithm will never discard a request. Instead, *CLK* will always generate the same set of dummy requests for mobile requests from the same mobile client. *CLK* maintain a 100% resilience against the corollary history attack in continuous queries under any spatial tolerance demanded by the mobile client. For the algorithms *B*, *T*, *H* and *Py*, an increase in spatial tolerance implies stronger resilience against corollary attacks. In general, as the spatial tolerance increases, it becomes easier to locate the same mobile clients that were used as previous anonymization candidates.

Additionally, *CK* also achieves high resilience when the spatial tolerance is very low (Qos high). This is attributed to the fact that under low spatial tolerances (50m * 50m) the privacy requirement (K=10) may be difficult to achieve.

However, instead of discarding the request, *CK* will generate the same set of dummy requests, similar to *CLK*. The main difference between *CLK* and *CK* is that *CK* will first make an attempt to anonymize with real mobile requests. However, if there is limited real mobile requests, *CK* generate the same set of dummy requests for mobile requests from the mobile client. On the other hand, *CLK* does not search for real mobile requests. Instead, *CLK* quickly generates the same set of dummy requests for mobile requests from the same client.

8.15.8 Resilience and global privacy requirement

In Figure 67, we evaluated the effects of K on the resilience of the algorithms.

We configured *profileCount*=10, this implies that the proposed algorithms (*CLK*, *CK*) will not generate more than 10 realistic diverse dummy requests not exceeding K =10 from the dummy profile (*see Section 8.6.1*).

For this study, 5000 mobile requests where submitted and the average spatial tolerance is 200m * 200m. First, it is observe from Figure 67 that for very low values of *K* (e.g. *K*=2), all the algorithms provided a high level of resilience to the corollary attacks. In general, as K increases the resilience of the algorithms decrease. As *K* increases, it becomes difficult to locate the previous anonymized candidates. For the *CLK* algorithm, we observe that as the privacy requirement (i.e. *K*) surpasses *profileCount*, the resilience decreases (e.g. at *K*=12, *K*=14). It therefore makes sense in the *CLK* algorithm to ensure that the *profileCount* system parameter setting is configured to a large value.

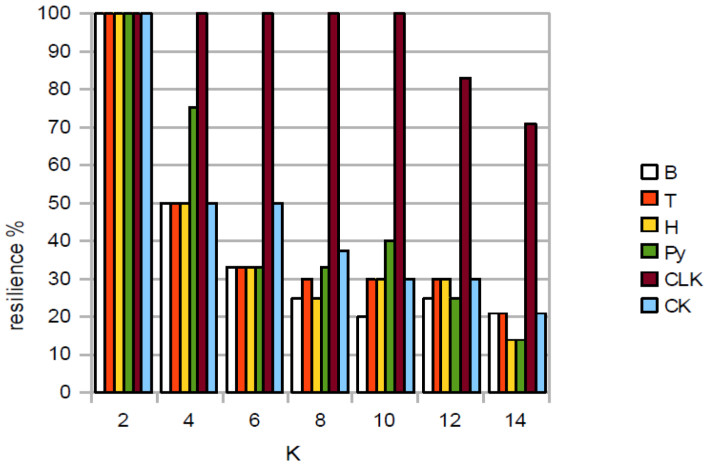


Figure 68-Resilience and global privacy

8.16 CloakLessK and CloakedK

We introduced the *MobiPriv* suite of algorithms. Both algorithms *CloakLessK* and *CloakedK* are able to achieve the maximum (100%) success rate under any personalized or system wide anonymity level or spatial resolution. This is not the case for previously proposed solution to the same problem. Instead, under certain circumstances as depicted by the experiments, these previously proposed models will underperform.

Both algorithms (i.e. CLK and CK) can guarantee a very high success rate. In terms of run time performance, CLK outperforms CK since there is no search for K-1 other requests in CLK. Instead, CLK can quickly generate K-1 other fake requests.

With regards to communication cost, *CloakedK* outperforms *CloakLessK*. Moreover, we saw that the communication cost of *CloakLessK* is not affected by spatial resolution unlike *CloakedK*. The communication cost of *CloakedK* improves as the anonymity level increases or if the spatial tolerance increases. *CloakedK* does much better than *CloakLessK* as the number of request sent to our middleware increases.

CloakLessK is effective for privacy preservation in continuous queries. The resilience of *CLK* is not affected by the number of continuous snapshot requests sent or the quality of service demanded by the mobile client. *CLK* offers very high resilience to corollary attacks even under increasing privacy requirement (*K*) by the mobile user. When the privacy requirement of the mobile client surpasses *profileCount*, the resilience is reduced. It is therefore recommended that the parameter *profileCount* be a very large value. For example, *profileCount* could be the maximum *K* value allowed in the system.

In summary, the proposed algorithms are simple yet effective for privacy preservation in snapshot and continuous querying systems. Additionally, we satisfied both location and query privacy.

8.17 Discussion

In this dissertation, we propose and evaluate a suite of privacy preserving algorithms for location based systems called MobiPriv. MobiPriv is compared against previously proposed privacy preservation algorithms for mobile location based systems [22, 9]. Results indicate that CLK and CK substantially improved the success rate in location based privacy systems. Even with high quality of service requirement, CLK and CK can achieve high success rate. With previously proposed algorithms, under certain conditions, such as high quality of service requirement, or high anonymity level, these privacy systems discard a number of mobile requests that cannot be anonymized. On the other hand, MobiPriv provides a “guaranteed service“, and all the mobile requests sent to our anonymization server can be anonymized. There is also no temporal delay for *K-1* other clients to become available in the proposed work.

Further empirical evaluation showed that *MobiPriv* algorithms have a good run time performance. In particular, for low anonymity level, MobiPriv's *CK* supersedes the other privacy algorithms such as Casper [9] and PrivacyGrid [22] in run time performance. This run time result conforms to the algorithm complexity asymptotic analysis. The communication cost of MobiPriv algorithms is higher. However, a communication cost reduction strategy is used in MobiPriv's *CK*.

Using a dynamic dummy generation strategy that preserves the K-anonymity and diversity requirement, *CLK* and *CK* is effective in a continuous querying environment and guarantees resilience against query linking via corollary history attack. The experimental results indicate that MobiPriv's *CLK* is the best privacy in continuous querying environment. These results also indicate that previously proposed snapshot privacy algorithms, such as Casper and PrivacyGrid cannot overcome the privacy challenges in continuous queries. In general MobiPriv is effective under everyday usage, it guarantees that all queries are successfully anonymized in a fast time.

9. CONCLUSION AND FUTURE WORK

We proposed a new robust approach to detecting outdoor transportation modes using a single smartphone sensor (i.e. GPS). In the proposed work, we considered and used transportation network data consisting of real time location of buses, rail lines, and bus stops spatial data. The real time location of buses is available in many cities such as Chicago, New York, Toronto, London, Washington DC, and San Francisco. Using the transportation network data, we showed that it is possible to address the weakness of previously proposed solutions [11, 12, 13, 14, 15]; that is, to distinguish between motorized modes such as trains, buses, and cars with high accuracy. Furthermore, if we detect that a traveler is traveling by bus, we can further identify on which particular bus the person is traveling.

Among the five classification models considered, Random Forest model is the most dominating classification model with over 93 % precision and recall accuracy. When transportation network classification features are not considered, the precision accuracy decreased to below 76%. This reduction of accuracy, upon omission of transportation network related features, is more notable for motorized transportation modes and bikes. This implies that transportation network data is effective for detecting motorized transportation, and bikes.

We also realized that, in order to achieve high precision and recall accuracy for outdoor transportation mode detection, only a subset of our initial set of classification features is necessary. In addition to traditional features on average speed and average acceleration, we identified for the first time the features on average bus closeness, average rail line closeness, and average candidate bus closeness. Using only this subset of features, and suppressing the other classification features that are not necessary, the precision accuracy was still over 92.5%.

For indoor and outdoor transportation mode detection, we added the accelerometer sensor to the GPS model. Combining these two sensors with geospatial knowledge increases the outdoor detection accuracy by over 5.5% compared to using only GPS and GIS as done in our prior work [8]. The use of several meta-level classifiers such as stacking and voting formed by combining several base-level classifiers are found to be slightly more effective than the traditional base-level classifiers by themselves such as Decision Trees and Random Forest

Further evaluation reveals that the accelerometer sensor is more effective than GPS for indoor transportation mode detection. Using the accelerometer only, the system can still achieve over 95% detection accuracy. Using only the GPS sensor and geospatial knowledge of the transportation network such as real time bus location, the indoor transportation mode detection accuracy reduces to below 67% on average.

In terms of battery power consumption and sensor reports, experimental results on an IPhone 3G indicate that using both the GPS and accelerometer sensors exhaust the battery power much faster than using only accelerometer sensor reports. Thus, one could turn their GPS sensor off, conserve battery power, and still achieve a high level of accuracy for transportation mode detection service. Using only accelerometer reports and pruning GPS reports will reduce the mobile phone’s battery exhaustion rate by nearly 3 times.

For parking status detection, we propose cost effective solutions that utilizes mobile phone sensors such as GPS, accelerometer, and Bluetooth sensors. Furthermore, the parking status detection algorithm may piggy back on pay-by-phone for parking transactions.

Given the fact that not all drivers carry a mobile phone and the fact that not all drivers with a mobile phone have the PhonePark system installed, the penetration ratio of the PhonePark system may not be 100%. Additionally, PhonePark PSDs may produce false observations of parking activities due to GPS errors, transportation mode detection errors, and Bluetooth pairing errors. The parking availability estimation algorithms take these limitations into consideration and then provide a final estimate on the availability of street parking spaces.

Specifically, parking availability estimation (PAE) algorithms estimate the street parking availability in real-time by combining historical parking statistics with real-time parking observations. Several approaches on combining historical parking statistics with real-time parking observations are considered. Experimental and theoretical analysis demonstrate the effectiveness of the parking availability estimation algorithms. Specifically, for penetration ratios of 1%, the average error of street parking availability estimation is below 2. More studies show that the combination of historical parking statistics and real-time parking observations as done in weighted average (WA) and Kalman Filter (KF) is more effective than using solely historical parking statistics (HS) or solely real time observations (SPP).

In general, the proposed approach is economical, convenient, and flexible. It produces good estimation in real-time on the availability of street parking spaces. The estimation of parking availability has potential to improve drivers' parking experience by saving parking search time, reducing gas consumption and CO_2 emission.

Further, we propose and evaluate a suite of privacy preserving algorithms for location based systems called MobiPriv. MobiPriv is compared against previously proposed privacy preservation algorithms for mobile location based systems [9, 22]. Results indicate that MobiPriv's CLK and CK substantially improved the success rate in location based privacy systems. Even with high quality of service requirement, CLK and CK can achieve high success rate. With previously proposed algorithms, under certain conditions, such as high quality of service requirement, or high anonymity level, these privacy systems discard a number of mobile requests that cannot be anonymized. On the other hand, MobiPriv provides a “guaranteed service“, and all the mobile requests sent to our anonymization server can be anonymized.

In the future, we will study how different mobile phone orientations and placements, affect the transportation mode classification accuracy. In addition to mobile phone orientation studies, the conservation of battery power by selectively sampling the mobile phone sensors or adaptively controlling the sampling frequency is also a subject of future work. For example, since GPS does not contribute to indoor transportation mode detection accuracy, then when the traveler is indoors we can deactivate GPS.

Another area to explore in the future is the use of other sensors such as acoustic sensors for transportation mode activity and parking detection from smartphones. For example, the microphone can sense that the driver turned the engine off and did park. Improvements on parking availability estimation includes: (1) exploiting seasonal trends (e.g. weather) for historical availability profile construction, (2) aggregating historical information, prior parking observations in the previous time epoch, and current observations, and (3) utilizing parking and deparking signals from neighboring blocks when estimating parking availability for a candidate street block.

10. CITED LITERATURE

[1] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, P. Boda. PEIR, the Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research. ACM MobiSys, 2009.

[2] V. Manzoni, D. Maniloff, K. Kloeckl, C. Ratti. Transportation mode identification and real time CO_2 emission using smartphones. Technical Report, SENSEable City Lab, Massachusetts Institute of Technology, 2010.

[3] ImpactSports, 2008. ePulse - Armband Heart Rate Monitor. http://impactsports.com

[4] BodyMedia Sensewear wms. http://www.sensewear.com/images/WMSbrochure.pdf 2008.

[5] John Voelcker. Stalked by satellite: An alarming rise in GPS-enabled harassment. IEEE Spectrum, 2006.

[6] Leon Stenneth, Philip S. Yu, Ouri Wolfson. Mobile Systems Privacy: “MobiPriv” A Robust K-Anonymous System for Location Based Mobile Systems. 6th IEEE WiMob, 2010.

[7] Leon Stenneth, Philip S. Yu. Global Privacy and Transportation Mode Homogeneity Anonymization in Location Based Mobile Systems with Continuous Queries. 6th International Conference on Collaborative Computing: Networking, Applications and Work Sharing, 2010.

[8] L. Stenneth, O. Wolfson, P.S. Yu, B. Xu. Transportation Mode Detection from Mobile Phones and GIS Information. ACM SIGSPATIAL GIS, 2011.

[9] Mohamed Mokbel, Chi-Yin Chow, Walid Aref. The New Casper: Query Processing for Location based Services without Compromising Privacy. 32nd International Conference on VLDB, 2006.

[10] Chi-Yin Chow, Mohamed Mokbel, Xuan Liu. A peer to peer spatial cloaking algorithms for Anonymous Location Based Services. ACM GIS, 2006.

[11] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, M. Srivastava. Using Mobile Phones to Determine Transportation Modes. ACM Transactions on Sensor Networks, Vol. 6, No. 2, Article 13, 2010.

[12] L. Liao, D. Fox, H. Kautz. Learning and Inferring Transportation Routines. AAAI, 2004.

[13] D. Patterson, L. Liao, D. Fox, and H. Kautz. Inferring High-Level Behavior from Low-Level Sensors. ACM UBICOMP, 2003.

[14] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W. Ma. Understanding mobility based on GPS data. In Ubiquitous Computing. ACM New York, 2008.

[15] Y. Zheng, L. Liu, L. Wang, X. Xie. Learning Transportation Mode from Raw GPS Data for Geographic applications on the Web. WWW, 2008.

[16] A. Guttman. R-trees: a dynamic index structure for spatial searching. ACM SIGMOD, June 1984.

[17] San Francisco Park website http://sfpark.org/.

[18] Parking Carma website, http://www.parkingcarma.com.

[19] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe. ParkNet: drive-by sensing of road-side parking statistics. ACM MobiSys, 2010

[20] Parkmobile's website. http://us.parkmobile.com/

[21] Parkmobile's Pay by Phone website. http://us.parkmobile.com/en/everyone/how-it-works/pay-by-phone-parking

[22] Bhuvan Bamba, Ling Liu, Peter Pesti, Ting Wang. Supporting anonymous Location Queries in Mobile Environments with PrivacyGrid . World Wide Web, 2008.

[23] E. Murakami, and D. P. Wagner, D. Neumeister. Using Global Positioning Systems and Personal Digital Assistants for Personal Travel Surveys in the United States," International Conference on Transport Survey Quality and Innovation, July 2004.

[24] E. Murakami, and D. P. Wagner. Can using global positioning system (GPS) improve trip reporting? Transportation Research Part C, 7(2/3), 1999.

[25] P. Gonzalez, J. Weinstein, S. Barbeau, M. Labrador, P. Winters, N. Labib, R. Perez. Automating mode detection using neural networks and assisted GPS data collected using GPS enabled mobile phones. 15th World Congress on Intelligent Transportation Systems, 2008.

[26] S. Consolvo, D. Mcdonald, T. Toscos, M. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. Lamarca, L. Legrand, R. Libby, I. Smith, and J. Landay. Activity sensing in the wild: A field trial of UbiFit garden. 26th Annual Conference on Human Factors in Computing Systems, 2008.

[27] Apple Computer, Inc., Nike+ iPod Sport Kit. http://apple.com/nike Nike, 2008

[28] L. Stenneth, O. Wolfson, B. Xu, P. S. Yu. Street parking using mobile phones. IEEE MDM, 2012.

[29] J. Farringdon, A. Moore, N. Tilbury, J. Church, P. Biemond. Wearable sensor badge & sensor jacket for context awareness. In Proceedings of the 3rd International Symposium on Wearable Computers, 1999.

[30] N. Kern, B. Schiele, and A. Schmidt. Multi-sensor activity context detection for wearable computing. In Lecture Notes in Computer Science, Springer-Verlag, 2003

[31] L. Bao, S. Intille. Activity recognition from user-annotated acceleration data. In Lecture Notes in Computer Science. Springer-Verlag, 2004.

[32] J. Lester, T. Choudhury, G. Borriello. A practical approach to recognizing physical activities. In Lecture Notes in Computer Science, vol. 3968, Springer-Verlag, 2006.

[33] J. Froehlich, T. Dillahunt, P. Klasnja, J. Mankoff, S. Consolvo, B. Harrison, J. Landay, UbiGreen: Investigating a mobile tool for tracking and supporting green transportation habits. Computer Human Interaction. ACM, New York, 2009.

[34] T. Sohn, A. Varshavsky, A. Lamarca, M. Chen, T. Choudhury, I. Smith, S. Consolvo, J. Hightower, W. Griswold, E. De Lara. Mobility detection using everyday GSM traces. In Lecture Notes in Computer Science, vol. 4206, Springer-Verlag, 2006.

[35] M. Ermes, J. Parkka, J. Mantyjarvi, I. Korhonen. Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions. IEEE Transactions on Information Technology in Biomedicine, 2006.

[36] J. Parkka, M. Ermes, J. Mantyjarvi, I. Korhonen, J. Peltola. Activity classification using realistic data from wearable sensors. IEEE Transactions on Information Technology in Biomedicine, 2006.

[37] M. Oliver, M. Badland, S. Mavoa, M. Duncan. Combining GPS, GIS, and Accelerometry: Methodological Issues in the Assessment of Location and Intensity of Travel Behaviors. Journal of Physical Activity and Health, 2010.

[38] N. Schüssler, K. Axhausen. Processing GPS raw data without additional information. Journal of the Transportation Research Board, 2009.

[39] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, K. Aberer. SeMiTri: A framework for semantic annotation of heterogeneous trajectories. EDBT, 2011.

[40] P. Troped, M. Oliveira, C. Matthews, E. Cromley, S. Melly, B. Craig. Prediction of Activity Mode with Global Positioning System and Accelerometer Data. Journal of Medicine & Science in Sports & Exercise, vol. 40(5), 2008.

[41] H. Lu, J. Yang, Z. Liu, N. Lane, T. Choudhury, A Campbell. The Jigsaw Continuous Sensing Engine for Mobile Phone. ACM SenSys, 2010.

[42] Y. Wang, J. Lin, M. Annavaram, Q. Jacobson, J. Hong, B. Krishnamachari, N. Sadeh. A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition. ACM MobiSys, 2009.

[43] P. Zhou, Y. Zheng, M. Li. How long to wait?: Predicting bus arrival time with mobile phone based participatory sensing. ACM MobiSys, 2012.

[44] S. Lin , Y. Chen, and S. Liu. A Vision-Based Parking Lot Management System. IEEE International Conference on. Systems, Man and Cybernetics, 2006.

[45] H. Al-Absi, J. Devaraj, P. Sebastian, and Y.V. Voon .Vision-based automated parking system. ISSPA, 2010.

[46] Smart-parking at rockridge BART station, http://www.path.berkeley.edu/path/research/featured/120804/smart-park.html.

[47] About Airport Parking website, http://www.aboutairportparking.com/

[48] H. Kurogo, K. Takada, and H. Akiyama. Concept of a parking guidance system and its effects in the Shinjuku area-configuration, performance and future improvement of system. Proceedings of Vehicle Navigation and Information Systems Conference, 1995.

[49] D. Ayala, O. Wolfson, B. Xu, B. DasGupta and J. Lin, Stability of Marriage and Vehicular Parking, 2nd International Workshop on Matching Under Preferences, 2012.

[50] D. Ayala, O. Wolfson, B. Xu, B. DasGupta and J. Lin, Parking in Competitive Settings: A Gravitational Approach, IEEE MDM, 2012

[51] D. Ayala, O. Wolfson, B. Xu, B. Dasgupta and J. Lin, Parking Slot Assignment Games, 19th ACM GIS, 2011

[52] E. Kokolaki, M. Karaliopoulos, I. Stavrakakis. Opportunistically-assisted parking service discovery: now it helps, now it does not. Pervasive and Mobile Computing, Elsevier, vol. 8(2), 2012.

[53] E. Kokolaki, M. Karaliopoulos, I. Stavrakakis. Value of information exposed: wireless networking solutions to the parking search problem. 8th International Conference on Wireless On-demand Network Systems and Services (IFIP/IEEE WONS), 2011

[54] E. Kokolaki, M. Karaliopoulos, I. Stavrakakis. Trading public parking space. Department of Informatics and Telecommunications. University of Athens, 2012

[55] E. Kokolaki, M. Karaliopoulos, I. Stavrakakis. Leveraging information in vehicular parking games. Technical Report, Department of Informatics and Telecommunications. University of Athens, 2012

[56] Google Open Spot Parking http://techcrunch.com/2010/07/09/google-parking-open-spot/

[57] Google's Open Spot for street parking availability. http://www.androidauthority.com/google-labs-open-spot-a-useful-application-that-no-one-uses-15186/

[58] Pierangela Samarathi, Latanya Sweeney. Protecting Privacy when disclosing Information: k-Anonymity and its Enforcement through Generalization and Suppression. SRI-CSL-98-04.

[59] Marco Grusteser, Dirk Grunwald. Anonymous usage of location based services through spatial and temporal cloaking. ACM/USENIX MobiSys, 2003.

[60] Bugra Gedik, Ling Lui. Location Privacy in Mobile Systems: A Personalized Anonymization Model. ICDS, 2005.

[61] Man Yiu, Christian Jensen, Xuegang Huang, Hua Lu. SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. 24th International Conference on Data Engineering , 2008.

[62] Hidotoshi Kido, Yutaka Yanagisawa, Tetsuji Satoh. An Anonymous Communication Technique using Dummies for Location Based Services. Second International Conference on Pervasive Services, 2005.

[63] Tun-Hao You, Wen Peng, Wang Lee. Protecting Moving Trajectories Using Dummies. International Workshop on Privacy-Aware Location-Based Mobile Services, 2007.

[64] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, Kian Tan. Private queries in Location Based Services: Anonymizers are not Necessary. SIGMOD, 2008.

[65] Chi-Yin Chow, Mohamed Mokbel, Tian He. Tiny Casper: A Privacy-Preserving Aggregate Location Monitoring System in Wireless Sensor Networks. SIGMOD, 2008.

[66] Chi-Yin Chow, Mohamed Mokbel. Enabling Private Continuous Queries for Revealed User Locations. International Symposium on Advances in Spatial and Temporal Databases, 2007.

[67] Walk Jog Run. http://www.walkjogrun.net/

[68] Mountain Bike. http://www.mtbroutes.co.uk/northyorkmoors/default.aspx

[69] SportsDo. http://sportsdo.net/Activity/ActivityBlog.aspx

[70] Wikiwalki. http://www.wikiwalki.com

[71] Chicago Transit Authority Bus Tracker http://www.ctabustracker.com/

[72] J. Hellerstein, J. Naughton, and A. Pfeffer. Generalized Search Trees for Database Systems. 21st International Conference on Very Large Data Bases, 1995.

[73] Yang, T. Toida, C. Hong. Transportation prediction using build-in triaxial accelerometer in cell phone. International Conference on Business Information, 2010.

[74] N. Ravi, N. Dandekar, P. Mysore, M. Littman. Activity recognition from accelerometer data. AAAI, 2005.

[75] L. Kuncheva. Combining Pattern classifiers: Methods and Algorithms. John Wiley and Sons, Inc, 2004.

[76] L. Breiman. Bagging Predictors. Machine Learning, 1996.

[77] M. Mathie. Monitoring and interpreting human movement patterns using a triaxial accelerometer. PhD Thesis. New South Wales, 2003.

[78] J. Eriksson, L. Girod, B. Hull. The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. ACM MobiSys, 2008.

[79] I. Witten, E. Frank. Data Mining: Practical machine learning tools and techniques. Morgan and Kaufmann, San Francisco, 2005.

[80] R. Duda, P. Hart, D. Stork. Pattern Classification. Wiley, New York, 2000.

[81] R. Meir and G. Ratch. An introduction to boosting and leveraging. Advanced Lectures Notes on Machine Learning (LNCS), 2003.

[82] D. Wolpert. Stacked generalization. Neural Networks, 1992.

[83] I. Guyon, A. Elisseef. An Introduction to Variable and Feature Selection. Journal of Machine Learning Research, 2003.

[84] Monsoon Power Monitor for mobile device. http://www.msoon.com/LabEquipment/PowerMonitor/

[85] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Grusteser, R. Martin. Sensing Vehicle Dynamics for Determining Driver Phone Use. ACM MobiSys 2013.

[86] Tingxin Yan, Baik Hoh, Deepak Ganesan, Ken Tracton, Toch Iwuchukwu, and Juong-Sik Lee. A crowdsourcing-based parking reservation system for mobile phones. Technical Report UM-CS-2011-001, Department of Computer Science, University of Massachusetts, Amherst MA, 2011.

[87] S. Nawaz, C. Efstratiou, C. Mascolo. ParkSense: A Smartphone Based Sensing System For On-Street Parking. ACM MobiCom, 2013.

[88] V. Corric, M. Gruteser. Crowdsensing Maps of On-Street Parking Spaces. IEEE International Conference on Distributed Computing in Sensor Systems, 2013.

[89] K.A. Myers and B.D. Tapley. Adaptive Sequential Estimation with Unknown Noise Statistics. IEEE Transactions on Automatic Control, Vol. 21, No. 4, pp.520-523, 1976.

[90] L. Chu, S. Oh, and W. Recker. Adaptive Kalman filter based freeway travel time estimation. TRB Annual Meeting, Jan., 2005.

[91] USA Today. Authorities: GPS system used to stalk woman. http://www.usatoday.com/tech/news/2002-12-30-gpsstalkerx.htm

[92] Fox News. Man Accused of Stalking Girlfriend with GPS. http://www.foxnews.com/story/0,2933,131487,00.html

[93] Anonymous surfing. http://www.anonymizer.com

[94] L. Stenneth, P. S. Yu. Mobile Systems Privacy. "MobiPriv" A Robust K-Anonymous System For snapsot and continious querying location based systems. Transactions on Data Privacy 2011

[95] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, Muthuramakrishnan Venkitasubramaniam, “L-diversity”: Privacy beyond k anonymity. ICDE, 2006.

[96] Latanya Sweeney. K-anonymity: A model for Protecting Privacy. International Journal on Uncertainty and Fuzziness and Knowledge Based Systems, 2002.

[97] Ling Lui. From Data Privacy to Location Privacy: Models and Algorithms. VLDB, 2007.

CURICULUM VITAE

Experience

2013 - Present. Nokia Connected Driving Chicago, IL

Research Scientist (Full time)

2008 - 2013. University of Illinois Chicago, IL

Research Assistant while doing PhD

Professor Philip Yu: context mining, social networks analysis, and privacy preservation

Professor Ouri Wolfson: mobile computing, spatio-temporal databases, and ITS

Professor Jakob Eriksson: Sensor network, GIS, mobile handheld, and transportation systems

2011 - 2012 Computerized Facilities Integration Chicago, IL

Software architect (contract 1YR) – Java based on the Archibus space management system (http://www.archibus.com/).

Agile development with scrum methodology. Tools used: Java, Oracle database, scripting

2011 NAVTEQ (acquired by Nokia) Chicago, IL

Software/Research consultant (contract 6 months)

Automatic software testing of map objects. Given new maps, I developed a strategy to automatically verify if mapping operations such as panning and zooming are supported.

Tools used – Robot, Selenium IDE, Selenium RC, Web Driver, Java, JUNIT, Maps (Navteq, Bing, OpenLayers), Tomcat, SmartGWT.

2010-2011 Research In Motion (BlackBerry) Rolling Meadows, IL

Software developer (intern 4 months)

Chief architect and developer for the DunnsRiver Intelligent Patent Management

System. This system simplifies the search process for patents and can also recommend areas for patenting.

Techniques used – Text mining, JEE tools, Hibernate Core/Search, Web Services (Axis-SOAP), Wiki, Apache Tomcat, Drools, Maven, PostgreSQL, Oracle, Bugzilla, Carrot 2.

2009 – 2010 BYTELAW Golf Road, IL

Software consultant (contract)

Patent extraction and search enterprise system. Tools used: Java, JEE, REST, MySQL

2004–2008 University of Technology Jamaica Kingston, Jamaica

Lecturer

2006–2008 Bank of Jamaica Kingston, Jamaica

Software consultant

Consultancy on their Java based Financial System.

Education

2008 – Fall of 2013 University of Illinois Chicago, IL

PhD Computer Science – Current GPA = 3.77/4.00

Thesis – Detecting human activities using smartphones and maps.

Advisors: Professor Philip Yu and Professor Ouri Wolfson

2003–2004 University of Westminster London, UK

Master of Science – Distinction, GPA = 4.00

Double Specialization: Applied Programming and Computer Networks

1999–2003 University of Technology Jamaica Kingston, Jamaica

BSc Computer Science – Distinction, GPA = 3.77/4.00

Technical skills

Java Standard Edition (JSE), Java Enterprise Edition (JSP,JSF, Java Beans, Servlets, JDBC), XML, JSON, Hibernate Core/Search, AXIS(SOAP) ,SEAM, C++, C, C#, J2ME, DROOLS, MAVEN, Databases (MS SQL, MYSQL, PostgreSQL, Oracle) VB, desktop website design, mobile web development, Bugzilla, Wiki Programming, AJAX, SJAX, JSON, Mobile handheld development, Application Servers (Apache Tomcat, JBoss, Jetty), Maps API (Google, Bing, Esri), Geo-location Apps, GIS applications, Map matching, Network Programming, Team Foundation Server, Context Aware Applications, Automated Software testing (Selenium, Robot, JUNIT), Agile and Scrum development. Machine learning (Classification, Clustering, Feature Selection)

Popular Project/s

TransitGenie – A mobile web and IPhone context aware transit itinerary planner for the city of Chicago. http://www.transitgenie.com/

PhonePark – Street parking using mobile phones (IEEE MDM 2012)

Awards

Most Enthusiastic Inventor (Nokia 2013)

Chevron International Student Scholarship 2011

Richard Tapia Diversity Fellowship 2011

Fulbright Scholarship for PhD 2008

European Union Scholarship for MSc 2004

University of Technology Foundation Scholarship for 4 years of BSc 1999-2003

Selected publications

Journal

Mobile Systems Privacy. "MobiPriv" A Robust K-Anonymous System For snapsot and continuous querying location based systems.
Transactions on Data Privacy 2011
Leon Stenneth, Philip S. Yu, Ouri Wolfson

Conference

Spatio Temporal Matching Under Data Uncertainty. *30th IEEE ICDE 2014 (under review)*.
Daniel Ayala, Ouri Wolfson, **Leon Stenneth**, Bashkar Dasgupta, Jane Lin

BATE: Bus arrival time estimation in a timetable environment. *30th th IEEE ICDE 2014 (under review)*
Leon Stenneth, Philip Yu

Real time street parking availability estimation. *13th IEEE MDM 2013*
Bo Xu, Ouri Wolfson, Leon Stenneth, Jie Yang, Pete Nelson

Activity recognition from smartphone sensor traces and geospatial knowledge. *Springer Distributed and Parallel Databases 2013 (under review)*
Leon Stenneth, Ouri Wolfson, Philip Yu, Bo Xu

Monitoring and Mining GPS traces in transit space. SIAM SDM 2013
Leon Stenneth, Philip Yu

An Empirical study of Data Race Detector tools. International conference on soft computing 2013
Jalal Alowibdi, Ugo Buy, **Leon Stenneth**

Transportation transfers from GPS traces. 15th IEEE ITSC 2012
Leon Stenneth, Kenville Thompson, Waldin Stone, Jalal Alowibdi

PhonePark: Street parking using mobile phones. 12th IEEE MDM 2012
Leon Stenneth, Ouri Wolfson, Bo Xu, Philip Yu

Detecting human activities from smartphones and maps. SIAM SDM 2012 (Poster)
Leon Stenneth, Philip Yu, Ouri Wolfson, Bo Xu

Transportation mode detection using mobile devices and GIS. 19th ACM SIGSPATIAL GIS 2011
Leon Stenneth, Ouri Wolfson, Philip Yu, Bo Xu

Privacy-Aware Mobile Location Based Systems. 13th ACM UBICOMP 2011 (MLBS)
Leon Stenneth, Philip Yu

Reducing travel time by Incident reporting via crowdsourcing. 12th ICOMP 2011

Leon Stenneth, Waldin Stone, Jalal Alowibdi

Mobile Systems location privacy. “MobiPriv” A robust K-anonymous system for snapshot and continuous queries. Transactions on Data Privacy volume 5, issue 1.
Leon Stenneth, Philip Yu

Mobile Systems Location Privacy: "MobiPriv" A Robust K Anonymous System. 6th IEEE WiMob 2010
Leon Stenneth, Philip Yu, Ouri Wolfson

Global privacy and transportation mode homogeneity anonymization in location based mobile systems with continuous queries. *6th IEEE ColoborateCom 2010*
Leon Stenneth, Philip Yu

An Application of Intelligent Adaptive Agents Over Wireless Networks. *International Conference on Information and Communications Technology Research, 2009*
Leon Stenneth, Waldin Stone

A Peer Based Model for 2D Multiplayer Gaming. *4th International Conference on Internet Technologies and Applications ITA 2005*
Leon Stenneth, Richard Pyne

Workshop

Supporting Privacy and Auditing in Cloud Computing Systems, 2013.
Proceedings of the 1st IEEE International Workshop on Cloud Security Auditing 2013
Tyrone Grandison, Sean Thorpe, **Leon Stenneth**

Patents pending

Software system and methods for transforming probe data across transportation modes
Inventor: Leon Stenneth

Turn restriction determination. **US serial# 14/015185**
Inventor: Leon Stenneth

System and methods for matching transit vehicles to trips in a real time transit tracking environment.**US serial # 14/012329**
Inventor: Leon Stenneth

System and method for advanced selection of a resource among available resources.
Inventors: Ouri Wolfson, **Leon Stenneth** ,Philip Yu, and Bo Xu

Systems and methods for location proof beyond location positioning **US serial# 13/951740**

Inventors: Leon Stenneth

Systems and methods for automated street parking payment detection using mobile device and a map. **US serial# 13/915230**
Inventors: Leon Stenneth

Vehicle arrival prediction **US serial# 13/849012**
Inventors: Leon Stenneth, Leo Modica

Structure access characteristics determined from mobile unit data**. US serial# 13/784120**
Inventors: Leon Stenneth, Leo Modica

Systems and methods for activity classification, 2012 - **US serial# 61/720,815** highlight
Inventors: Leon Stenneth, Ouri Wolfson, Philip Yu, and Bo Xu

Methods and apparatus for transit mapping, 2012 - **US serial# 13/780901**
Inventors: Leon Stenneth, Leo Modica

Methods and apparatus for deriving spatial properties of bus stops and traffic controls, 2012 - **US Serial # 13/872439**
Inventors: Leon Stenneth, Leo Modica

Real-time vehicle spacing control, 2012 **- US serial# 13/739709**
Inventors: Leon Stenneth, Leo Modica

Deviation detection in mobile transit systems, 2012 - **US serial# 13/739560**
Inventors: Leon Stenneth, Leo Modica

Methods and apparatus for transit mapping, 2012 **- US serial# 13/781188**
Inventors: Leon Stenneth, Leo Modica