Mobile crowdsensing applications for intelligent parking and mobility

Cherian, Jim

2019

Cherian, J. (2019). Mobile crowdsensing applications for intelligent parking and mobility.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/106466>

<https://doi.org/10.32657/10220/47944>

Mobile Crowdsensing Applications for Intelligent Parking and Mobility

JIM CHERIAN

School of Computer Science and Engineering
Nanyang Technological University, Singapore

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy (Ph.D)

2019

To my family

and

all my teachers

Statement of Originality

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

Jim Cherian

01 February 2019

Supervisor Declaration Statement

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

Associate Professor Luo Jun

01 February 2019

Authorship Attribution Statement

This thesis contains material from 3 paper(s) published in the following peer-reviewed conference proceedings and journals where I was the first and corresponding author.

Major parts of Chapter 3 are published as Jim Cherian, Jun Luo, Hongliang Guo, Shen-Shyang Ho, and Richard Wisbrun. ParkGauge: Gauging the Occupancy of Parking Garages with Crowdsensed Parking Characteristics. In Proceedings of the 17th IEEE Conference on Mobile Data Management (MDM'16), pp. 92-101, Porto, Portugal, June 2016. DOI: 10.1109/MDM.2016.26. A preliminary version of this work in Chapter 3 was also presented in ACM SenSys 2015 and published as Jim Cherian, Jun Luo, Hongliang Guo, Shen-Shyang Ho, and Richard Wisbrun. Poster: ParkGauge: Gauging the Congestion Level of Parking Garages with Crowdsensed Parking Characteristics. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys'15), pp. 395-396, Seoul, South Korea, November 2015. DOI: 10.1145/2809695.2817881.

The contributions of the co-authors are as follows:

- Assoc Prof Luo Jun provided the project direction, carefully guided me throughout the development of methodology and edited the manuscript drafts.
- I prepared the manuscript drafts. The manuscript was further revised by A/Prof Ho Shen-Shyang, Dr. Guo Hongliang and Dr. Richard Wisbrun.
- I co-designed the study with Assoc Prof Luo Jun and Asst Prof Ho Shen-Shyang. Dr. Guo Hongliang provided several technical directions while I developed the methodology. Dr. Richard Wisbrun helped to set out the project objective and

research problem, suggested several related works and put emphasis on the industrial application context while shaping up the research project.

- All the experimental evaluation based on the data collected from several volunteer drivers were conducted by me at and/or using resources provided by BMW-NTU Future Mobility Lab, Energy Research Institute @ NTU and CNCL, School of Computer Science and Engineering, NTU.
- Asst Prof Ho Shen-Shyang assisted in the data collection and provided guidance in processing, interpreting and mining the dataset for relevant information.

Chapter 4 is published as Jim Cherian, Jun Luo, and Shen-Shyang Ho. ParkLoc: Light-weight Graph-based Vehicular Localization in Parking Garages. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 2.3 (2018): 99. DOI: 10.1145/3264909. It was also presented during the premier conference UbiComp 2018 on October 10, 2018 at Singapore.

The contributions of the co-authors are as follows:

- Assoc Prof Luo Jun set the project direction, meticulously advised me throughout the research and empirical data analysis and edited the manuscript drafts.
- I wrote the drafts of the manuscript. I co-designed the study and revised the manuscript together with Assoc Prof Luo Jun and Asst Prof Ho Shen-Shyang.
- I conducted all the experimental evaluation based on the data (collected from several volunteer drivers) at and/or using resources provided by BMW-NTU Future Mobility Lab, Energy Research Institute @ NTU and CNCL, School of Computer Science and Engineering, NTU.
- Asst Prof Ho Shen-Shyang assisted in the data collection and provided guidance in processing, interpreting and mining the dataset for relevant information.

Jim Cherian
01 February 2019

Acknowledgement

After 4+ years of arduous learning at this temple of learning, probably nothing more can better summarize my state of mind than the famous Socratic Paradox “*I know that I know nothing*”. Having come thus far from a decade-long career in industry where I was rather undeservingly celebrated for the specific knowledge points that could win projects and train others, this has been a flame test for my ego and passion to learn. Yet this journey has helped me learn *how* to probe and gauge what *is* possible (deterministically), what *might* be possible (stochastically) and more importantly, what *is most likely not* possible (emphatically).

I’ve heard somewhere, prominently in religious contexts, that “the antidote to uncertainty is faith”. Bayesian learning has helped me only to reinforce this - that a strong belief or prior and the knowledge of conditional priors can work wonderfully towards reaching a useful estimate in decision making, even if exceptions are always to be expected the norm. I’ve also come to realize that “the norm” and “the normal” have different, rather subtle mathematical connotations, than those we may take for granted in everyday language and life. It is and has always been such realizations that help me value these years toiled on graduate studies. And, I am truly and deeply grateful to everyone and everything of help to me through this phase of life.

I thank my supervisor Prof. Luo Jun and co-supervisor Prof. Ho Shen-Shyang for their wholehearted and meticulous support, advices and guidance in forming concrete ideas for research, system implementation, data collection, experiments and evaluation methodology and scientific writing. I recall with immense gratitude that without them, I will not have reached the current stage of study and research.

I thank all my mentors from BMW Group (Munich) especially Dr. Richard Wisbrun and Ms. Heleen Wilmlink for their valuable advices in shaping up my research direction and funding the project. I thank all my co-authors and colleagues at NTU, specially Dr. Guo Hongliang, Dr. Cao Zhiguang, Dr. Sun Haoqi, Dr. Akshay Rao, Dr. Felix Klanner

and Dr. Yang Yan of BMW-NTU Future Mobility Research Lab and Dr. Yang Yanbing of CNCL for their valuable advices and camaraderie. I wholeheartedly thank Prof. Subodh Mhaisalkar, Executive Director of ERI@N, for his vision and help on research funding. I also thank all admin staff at ERI@N, CNCL and CeMNet for their support. I also thank my colleagues from my present workplace at CETRAN, NTU for their warm support and encouragement during the thesis examination process.

I thank all my friends for helping me in countless ways and especially to collect data for my research. I also thank all my ex-colleagues of a decade+ in industry - from Wipro, Harman and Delphi - for strengthening my software development skills and much more, that helped me navigate through obscure waters during the past 4+ years. I thank the numerous friends from the Mar Thoma Syrian Church of Singapore who helped my family and me countless number of times with their fellowship, love, support and guidance during all these years. I especially thank Reuben, Christin, John, Nithin, Thomas, Saji'chayan and family, Rajan'chayan and Jolly'ammama, Dinu'chayan and Preethi'ammamma, Lybu'chayan, Lynu'chayan, Shaji'chayan and their families amongst many others for being part of my extended family in Singapore. I also thank the Choir of MTSC-SG for the wonderful gift of harmonized music that helped me a lot to maintain balance in my life, and withstand the pressure that naturally comes packaged with graduate studies and research.

I recall all my teachers who taught me right from the basics till now, thanking them for the knowledge I could acquire and *learn by transfer* step by step, through my life. I acknowledge with gratitude and humility that I could come to write and present this thesis only by standing on the shoulders of giants.

I thank my family - especially my parents K V Cherian and Alice Mathew, parents-in-law George Easow and Annie George, sister Gem, Basil and Lizbeth, brother-in-law Sandy and Risha, my dear wife Sonia and dearest son David - for being a perennial source of strength & support all through my *belated* graduate student life (which has never been easy, neither for me nor for them), reinforcing me with their unconditional love, blessings and prayers despite all the valuable time I took away, that I could and should have spent with them.

Above all, I pray and thank God Almighty for gifting me with a valuable opportunity to study and research at NTU, helping me bridge my long academic gap of 12 years, and granting me good health, resources and time to focus on my research.

Abstract

Due to the worldwide rapid urbanization and rising private vehicular mobility, most cities face problems such as traffic congestion, parking deficit, and cruising for parking, leading to resource wastage and environmental costs. This includes the *first/last-furlong* problem for personal vehicular mobility in public places i.e. the challenge to find/locate a parking space near the destination. Intelligent mobile sensing, which fuses the computing/sensing capabilities of mobile devices with machine learning, holds a great potential towards solving such urban mobility challenges. *Opportunistic* Mobile Crowdsensing (MCS) has also emerged as a new paradigm that enables intelligent mobility applications and services. Mobile computing research has addressed the on-street/outdoor traffic and parking problems to a great extent. However, the occupancy estimation of indoor parking garages remains mostly unsolved - especially in garages that lack vehicle-tracking sensor infrastructure and/or real-time information delivery.

In fact, the major bottleneck that precludes us from reusing successful outdoor schemes into parking garages is the inadequate indoor localization potential, due to unavailability of infrastructural support such as GPS/WiFi especially when underground. In addition, enterprises using MCS face concerns regarding the scalability of their proven system deployments to newer venues; this typically calls for extra data collection for training/tuning, which can be prohibitively expensive to reach a certain spatial urban coverage required to bootstrap a viable service. Moreover, drivers who park their vehicles in such garages often face an embarrassing problem of forgetting where they parked. Most existing indoor solutions rely on infrastructure such as WiFi or BLE, whereas current smartphone-only proposals mostly require major data collection and training efforts per garage, making them impractical to adopt at indoor parking garages. In this thesis, we address these issues by proposing crowdsensing methods, implementing system prototypes and endorsing them through empirical evaluation.

In the first part of this thesis, we focus on the parking occupancy inference problem for indoor parking garages without precise localization. As we cannot use mobile crowd-

sensing for direct occupancy counting as in the existing proposals for outdoor parking, we propose ParkGauge+ as a scalable mobile crowdsensing system to *indirectly* infer the occupancy of parking garages from temporal parking characteristics reported by commodity smartphones, instead of counting the parked vehicles. ParkGauge+ adopts mostly low-power sensors (accelerometer, barometer, gyroscope) to detect driving states and driving contexts and infer parking characteristics, which are utilized to infer the occupancy using Supervised Learning (SL). To achieve a scalable performance and improved spatial coverage of urban deployments, we further propose to employ Transfer Learning (TL) methods to transfer the parking occupancy inference concepts from existing ParkGauge+ deployments. This minimizes any extra data collection efforts and costs for new installations. Through extensive experiments at commercial parking garages, we evaluate and demonstrate the potential of our methods that can achieve normalized error (NRMSE) of 0.09 and 0.15 respectively for SL and TL.

In the second part of this thesis, we directly address the indoor vehicular localization problem in GPS/WiFi-deprived environments such as indoor parking garages and propose a novel light-weight smartphone-only solution called ParkLoc. ParkLoc exploits the inherent planar graph structure of the navigable paths in parking facilities, to match a vehicle trajectory onto a sub-section of the map, by modeling them as sparse directed graphs. By posing the indoor parking localization task as a *subgraph isomorphism problem* and exploiting an approximate graph matching method, ParkLoc can track a vehicle in real-time with a median error of 4.8m and localize a parked vehicle with a median error of 2m from the nearest parking space. Furthermore, ParkLoc adopts the GraphSLAM algorithm from robotics research and applies it into the mobile sensing environment; it learns the map graph from the observed trajectory graphs and a given set of bootstrap (seed) landmark nodes, in a semi-supervised manner. A key benefit of our approach is that ParkLoc works off-the-shelf without any expensive on-site training or sensor data collection per garage. A comprehensive evaluation of ParkLoc through extensive experiments performed in 4 different garages reveals the promising performance of our graph-based approach for both localization and mapping.

Keywords

Intelligent Parking, Mobile Crowdsensing, Transfer Learning, Graph Matching, Vehicular Indoor Localization and Mapping.

Contents

Statement of Originality	iii
Supervisor Declaration Statement	iv
Authorship Attribution Statement	v
Acknowledgement	vii
Abstract	x
List of Figures	xv
List of Tables	xix
1 Introduction	1
1.1 Background	2
1.2 Scope of Research	4
1.3 Contributions	5
2 Literature Survey	6
2.1 Parking Occupancy Gauging	6
2.2 Mobile Crowdsensing	8
2.3 Context and Activity Recognition	9
2.4 Transfer Learning for Regression Tasks	9
2.5 Mobile Sensing for Indoor Localization and Mapping	10
2.6 Vehicular Localization and Mapping	11
2.7 Graph Matching	12
2.8 Summary	12

3	ParkGauge+: Towards Scalable Occupancy Inference for Parking Garages	13
3.1	Introduction	13
3.2	Overview of ParkGauge+	18
3.2.1	Design Rationale: Why Temporal Characteristics?	18
3.2.2	System Architecture	19
3.3	Methodology	22
3.3.1	Scalable Parking Occupancy Inference	22
3.3.2	Sensing Core of ParkGauge+	27
3.4	Empirical Evaluation	34
3.4.1	Data Collection and General Experiment Setup	34
3.4.2	Data Pre-Processing	35
3.4.3	Evaluating ParkGauge+ Sensing Core	37
3.4.4	Evaluating Parking Occupancy Inference	41
3.4.5	Evaluating Parking Occupancy Transfer Learning	45
3.5	Discussion	48
3.5.1	Minimal Data Requirements	49
3.5.2	Impact of garage restrictions	49
3.5.3	Impact of special events	50
3.6	Summary	50
4	ParkLoc: Light-weight Graph-based Vehicular Localization in Parking Garages	52
4.1	Introduction	53
4.2	Methodology	56
4.2.1	ParkLoc Architecture	56
4.2.2	Sensing: Preparation and Processing	58
4.2.3	Online Trajectory Grapher	59
4.2.4	Graph Matching and Localization	62
4.2.5	Mapper	68
4.3	Empirical Evaluation	71
4.3.1	Experimental Setup	71
4.3.2	Data Pre-processing	72
4.3.3	Graph-based Localization Accuracy	73

4.3.4	Mapping Performance	80
4.4	Discussion	84
4.4.1	Additional infrastructural support	84
4.4.2	Garage constraints	84
4.4.3	Multi-floor localization	85
4.5	Summary	85
5	Conclusion	86
5.1	Research Contributions	86
5.2	Future Directions	88
	References	91

List of Figures

3.1	Parking garages in the CBD area of Singapore, with percentages of the types of occupancy monitoring shown on the left. Garages with online occupancy information and on-street parking are marked in green and purple, respectively.	14
3.2	Queuing at indoor parking garages: (top) at entrance and (bottom) the interior	15
3.3	Parking availabilities for three close-by parking garages A, B and C during a day: even when A and C are both fully occupied (0 available spaces), B has plenty!	16
3.4	Illustration of temporal parking characteristics and driving contexts of a vehicle at an indoor parking garage.	18
3.5	Parking occupancy and time-to-park at a popular shopping mall: a negative correlation is obvious.	19
3.6	Architecture of ParkGauge+.	20
3.7	A basic regression model that can fit and recover the relation between parking occupancy and time-to-park. The red dashed line indicates a simple linear fit, whereas the blue solid line indicates a quadratic fit. . .	23
3.8	Barometer atmospheric pressure variations while changing floors (going downwards) in a parking garage with 3 floors underground.	29
3.9	Model of a parking process used by DCD.	30
3.10	Visualizing driving states by raw data.	32
3.11	(a) Evaluation of PAD performance as start/give-up-parking events estimated vs. actual at two parking garages CP1 and CP2 (b) CDF for error in time-to-park estimates.	37

3.12	Raw data traces corresponding to three parking trials with short, medium and long values of time-to-park . We provide full labels only to the long trace for better readability, but the feature-context correspondence can be easily discerned for the other two traces also.	38
3.13	Precision and recall for driving state detector.	40
3.14	Detection accuracy and latency for driving contexts driving , cruising , parked , queuing and floor-change under various window sizes W with a step size of 5s, followed by start-parking and give-up-parking under different geofence radii with a step size of 5m	42
3.15	Comparison of different regression methods	43
3.16	Comparison of ParkGauge+ performance with a simpler approach measuring time-to-park from GPS loss.	44
3.17	Determining the values of Pearson parameters ω and σ that yield the best performance for SVR(PUK). Parameter values on x-axis are uniformly spaced and clipped at 10 to enable a clearer presentation.	45
3.18	(1) green bars on the left indicate how the traditional learning methods perform on a small target training dataset D_T from a new garage, (2) blue bars on the right show transfer learning methods on a combined dataset D	46
3.19	Evaluation of <i>two-stage TrAdaBoost.R2</i> to find (a) the optimal value of parameter S i.e. the number of stage-2 iterations, and (b) the optimal base regression learner method, that could be adopted for deployment.	47
4.1	(left) Raw gyroscope data from 3 trips (different vehicles) at an indoor garage, following the same trajectory. Positive and negative peaks indicate left and right turns, respectively. (right) Histogram of the temporal distribution of turns in a trajectory graph, after pausing-adjustment. Labels a,b,c,d represent pose nodes where left/right turn maneuvers are made.	55
4.2	System architecture of ParkLoc.	56
4.3	Examples of map graph super-imposed on floor plan: (a) CP_4 , a large shopping mall parking garage (b) CP_2 , a small indoor parking garage in a university campus. We refer to Section 4.3.1 for details on the test sites.	63

4.4	Illustration of our modified VF2 algorithm for graph matching: (i) map graph with entry at node a and exit at node h; orange line indicates the actual trajectory path, (ii) observed noisy trajectory graph, (iii) trajectory graph with virtual edges for approximate matching, (iv) resulting matched trajectory graph with a virtual edge chosen over real edge, (v) resulting matched sub-graph of the map, (vi) DFS search-tree and table that illustrates the evolution of the state-space as per VF2; it eventually chooses state s_6 (green) representing sub-graph $a-b-c-f$ over s_5 (red) representing $a-b-c-d$ as the final match, since s_6 satisfies all the syntactical, semantic and spatio-temporal feasibility conditions while s_5 is spatio-temporally infeasible.	65
4.5	Illustration of GraphSLAM concepts. Map edges spatially constrain the red-colored pose nodes $\mathbf{x}_i$ based on noisy odometry with high uncertainty (illustrated via large circles); however, these nodes get iteratively optimized to $\mathbf{x}_i^*$ as the algorithm proceeds. Global landmark edges directly connect the origin $\mathbf{x}_0$ to yellow-colored <i>seeds</i> (λ_{glm}) with low uncertainty. Orange-colored <i>intrinsic</i> landmarks (λ_{intr}) are automatically recognized from the planar structure and help optimize the map. Note that circles illustrating the uncertainty do not represent their actual distribution.	70
4.6	(a,c) Cumulative Distribution Function (CDF) plot of real-time tracking and parking localization error of ParkLoc vs. a Dead Reckoning approach, both using calibration parameter $k=0.35$ (b,d) Histogram showing the distribution of ParkLoc errors. Sub-figures (a,b) indicate the performance obtained with true, manually corrected map graphs, whereas (c,d) indicate the corresponding performance with maps auto-generated by ParkLoc mapper at the same parking garages.	74
4.7	CDF plot of real-time tracking error for different values of the calibration parameter k , when evaluated with (a) the true, manually corrected map graph and (b) auto-generated map graph.	75
4.8	Latency for (a,c) real-time tracking and (b,d) parking localization under various trajectory graph cut-off (subgraph) sizes. Sub-figures (a,b) uses the true, manually corrected map graphs whereas (c,d) use auto-generated map graphs.	76

4.9	Localization error vs minimum distance from end points of the localized map graph edge for (a) true manually corrected map and (b) auto-generated map. ParkLoc seems to work best when vehicle is in proximity of a map graph node.	76
4.10	CDF plot of real-time tracking error for different values of the trajectory subgraph (cut-off) sizes, using (a) the true, manually corrected map graph and (b) auto-generated map graph.	77
4.11	Choosing an optimal window size W_g for the vehicle motion model, that provides the best localization error and low latency, for (a) true manually corrected map and (b) auto-generated map.	78
4.12	Comparison of real-time tracking and parking localization performance between ParkLoc and related works.	79
4.13	Mapping performance at 4 parking facilities $CP1$, $CP2$, $CP3$ and $CP4$. (a-d) visualization of the learnt maps (red dotted lines) superimposed upon the true maps (blue lines), GPS trajectory if available (green lines), seeds and intrinsic landmarks (yellow and orange circles) (e) evaluation of mapping error vs number of seed nodes at different threshold turn angles θ_{thres}	81
4.14	Impact of landmarks (seed or intrinsic) on Mapper performance. Figure shows the number of landmarks required to achieve a minimum map quality in terms of RMSE (1, 2.5, 5 or 7.5 m) for CP1-4. Number of intrinsic landmarks detected remains fixed for each parking garage and depends on its structure. For larger sized garages and/or those with a small number of uniquely distinguishable intrinsic landmarks, more seeds are required to achieve good performance.	83

List of Tables

3.1	List of states, contexts and parking characteristics.	21
3.2	Features used in classifying driving state.	31
3.3	Precision and recall results for the performance of driving state detector.	39
4.1	Description of parking facilities used for the empirical evaluation of ParkLoc	72

Chapter 1

Introduction

Rapid urbanization has been posing greater challenges on the mobility of the urban dwellers worldwide. Most cities find it challenging to cope with the demands and provide sufficient infrastructure to meet the mobility requirements of their residents. This includes the well-known first/*last-mile* problem [1] for public transport¹, as well as the increasingly important first/*last-furlong* problem² for personal vehicular mobility in public places i.e. the challenge to find/locate a parking space near the destination.

On the other hand, the proliferation of connected mobile devices such as smartphones and wearables have spawned a host of new mobility services and applications. By their very nature, mobile devices are particularly attractive to address the (sub-) problems relevant to mobility, since nearly all human users would carry these devices with them as they move. Mobile devices now possess progressively higher computing and sensing capabilities that allow them to perform *intelligent sensing*. Therefore, we can apply machine learning methods on mobile sensor data and learn models to infer the individual context of the mobile device and/or perceive the environment around it, besides monitoring the temporal evolution of these contexts. In short, we can train mobile sensor data-driven models to enable novel special-purpose mobility applications.

As this thesis unfolds, we present two clear-cut and complementary case studies of how such intelligent mobile sensing methods can be applied to help solve the first/last-

¹*last mile* is an informal phrase originally used in telecommunications, since the speed bottleneck in communication networks is mostly in the last mile or km of connectivity to end user; it has later been adopted into public transport networks with a similar implication i.e. the travel to a venue from the nearest high-speed train station by bus or walk is typically the speed bottleneck of the entire journey

²we define *last-furlong* in the same spirit as last-mile; a furlong is 1/8th of a mile, or roughly 201m

furlong urban vehicular mobility problems such as parking a private vehicle in particular. Given the paucity of thorough studies for the indoor parking problem, we primarily focus on indoor situations. The first study involves use of supervised machine learning as well as transfer learning methods to gauge the occupancy of large, indoor parking garages in a crowdsensing setting. The latter aims to locate the vehicle indoors, by means of a light-weight graph-based approach that utilizes only commodity smartphones of individual drivers and without relying on any special infrastructure. In fact, both of these studies derive motivation from the lack of sufficient indoor vehicular localization capabilities that has been one of the biggest bottlenecks towards adopting or reusing outdoor parking solutions into indoor parking garages. In this context, we hope and believe that this thesis will help accelerate the progress towards the large-scale adoption of infrastructure-free indoor-location-based applications and services that are powered by intelligent mobile sensing, crowdsensing and transfer learning.

1.1 Background

Spurred by urbanization and extreme growth in vehicle population, urban vehicular mobility infrastructure worldwide has been challenged towards over-utilization leading to traffic congestion and parking unavailability, especially so in mega-cities. In fact, vehicles cruising in search of on-street parking places have evidently led to serious traffic congestion [2] as well as add on to the non-productive economic costs of driving [3, 4]. In contrast, indoor multi-storey car parks (aka, parking garages), which are the major urban parking facilities for mega-cities, can also produce serious on-road congestion because of long queues of vehicles waiting to enter them, even though their aggregated total capacity may be sufficient to meet the parking demands. A reason for this paradox is that the real-time occupancy information measured at the garages (if any) is mostly displayed only locally, at the specific location or beside the accessor streets, rather than published online and available to drivers in advance. While this can be temporarily addressed by further enhancing the civic infrastructure or by deploying sensors on roads [5] or instrumenting a fleet of vehicles (as in ParkNet [6]), this is expensive and prohibitively so for developing economies. Nonetheless, the hidden costs of the urban parking problem such as the environmental impact in terms of emissions [7] cannot be discounted either.

Mobile computing has turned out to be a blessing towards solving the urban mobility challenges [8, 9]; through the careful application of machine learning and pattern recognition methods, it enables inexpensive yet efficient mobile sensing solutions and scalable mobile crowdsensing schemes (akin to “*a crowd of connected sensors on the move*”) [10]. Therefore, it will be very helpful if the crowdsensing ability of driving population can be *opportunistically* exploited to this end. In fact, ParkSense [11] and PocketParker [12] have recently demonstrated this for occupancy inference of on-street and outdoor area parking, taking support from WiFi and/or GPS which are unavailable to indoor parking garages. Moreover, they rely on counting the vehicles based on their detection of parking and unparking events, which is not very scalable for large-sized indoor garages. On the other hand, we cannot *directly* adopt human activity recognition or transportation mode/context detection methods [13, 14] to inertial sensor data and infer parking occupancy; instead, we will have to introduce additional indirect mechanisms to map the temporal sequence of the detected states/contexts (such as **turning** or **braking**) to the parking occupancy.

Although mobile crowdsensing is appealing as a powerful tool, it presents a few practical challenges that affect real deployments. A typical bottleneck is the demand for substantial data collection to train and/or tune the system for each new venue, as many of the existing proposals expect. However this is undesirable due to the economic costs and human efforts required. Classical machine learning methods may not be of direct help here, as they mostly expect that the entire data is derived from the same distribution. Such assumptions are hard to actualize for large, realistic systems that must support diversity. Therefore, it becomes imperative to develop clever adaptations or new methodologies to realize *scalable* crowdsensing schemes, that can minimize the expensive data collection efforts (inspired by the 3Rs paradigm viz., *reduce*, *reuse*, *recycle*) and are still able to serve new venues with acceptable performance.

Tracking a vehicle indoors or even locating a parked vehicle inside a large indoor parking garage while returning from a venue, represent another pressing first-furlong problem that drivers face. In fact, mobile sensing can come to the rescue even in this case. However, most of the existing indoor localization proposals assume the availability of WiFi or BLE and use received radio signal strength fingerprints for localization. Since these infrastructural supports are unavailable, we need to look for viable alternatives; dead reckoning based on inertial sensing works for short distance/time, but

errors rapidly grow due to their noise profile. VeLoc and VeTrack [15, 16] apply machine learning models to detect landmarks and improve the accuracy but they require significant training and complex parameter tuning efforts. Thus emerges the need for a more practical and light-weight vehicular localization scheme to help solve this important aspect of the last/first-furlong problem at indoor parking garages.

1.2 Scope of Research

In this thesis, we work towards addressing the above listed concerns and solving a few first/*last-furlong* mobility problems related to indoor parking garages and validating the findings through experimental evaluation. First of all, we investigate the possibility of adopting mobile crowdsensing to solve the indoor parking occupancy inference problem at parking garages. In particular, we attempt to answer the following sub-questions:

- Can we use mobile crowdsensing using only commodity smartphones to infer the occupancy of an indoor parking garage, without any infrastructural support?
- In the absence of any sensor-equipped infrastructural aids, can we identify and exploit any indirect indicators of parking occupancy?

Secondly, we consider the more general problem faced in mobile crowdsensing with regard to the scalability and sustainability of systems that are to be trained and tuned for individual venues. In particular, we consider the following sub-questions:

- Can we *scale* or *reuse* an existing crowdsensing system towards a viable deployment at a new venue, without major additional data acquisition?
- Can we apply the transfer learning methodology towards reusing and/or recycling the datasets required by crowdsensing schemes based on traditional supervised machine learning?

Thirdly, we look at the other important aspect of the last/first-furlong problem at indoor parking garages, i.e. indoor vehicular localization. Furthermore, we also aim to address the practical issues of unavailability or obsolescence of maps used for indoor localization. In particular, we consider the following sub-questions:

- Can we accurately estimate the indoor location of a vehicle in a parking garage using a localization method which is also *light-weight* and *infrastructure-free*?
- In the absence of any sensor-equipped infrastructural aids, can we identify and exploit any *indirect indicators* of the indoor location within a parking garage in order to perform localization and mapping?

1.3 Contributions

In a nutshell, we have made the following contributions in this thesis:

- We propose the novel idea of using time (as in *time-to-park* and *time-in-queuing*) as an indirect indicator of parking occupancy. We apply this idea to build ParkGauge+, a mobile system to infer the occupancy of large indoor garages, by employing a hierarchy of supervised machine learning methods that detect driving states and contexts from raw smartphone sensor data streams and their temporal sequences to derive parking characteristics that eventually help us infer occupancy. We evaluate these methods and study the influence of various parameters using real-life parking experiments at different indoor garages.
- We propose the novel idea of *transfer crowdsensing*, essentially the application of transfer learning to crowdsensing systems to accelerate their deployments at new venues, while reducing costs and efforts for data collection, labeling and training. We incorporate this sustainable and scalable mobile crowdsensing solution into our ParkGauge+ prototype and evaluate the performance improvements.
- We propose ParkLoc, a novel light-weight graph-based indoor vehicle localization system that can especially help to locate a parked vehicle by using inertial sensors that are typically deemed noisy and not useful for such applications. We exploit the inherent planar graph-structure of parking garages and the spatio-temporal characteristics of their turning points to implement our indoor vehicular tracking and localization methodology. We design ParkLoc in such a way that it can operate with both a true, predefined high-quality map graph or even simultaneously localize and map the navigable paths of the parking garage. To handle the issues of unavailability or obsolescence of maps without any infrastructure support, we implement a semi-supervised graph mapping scheme.

Chapter 2

Literature Survey

In this chapter, we closely review the existing proposals that employ mobile sensing for parking occupancy inference as well as indoor vehicular localization. In particular, we first show the mismatches between the existing literature on applying automated recognition of human activity and mobility contexts towards parking occupancy gauging (albeit plentiful in handling outdoor parking) in a mobile crowdsensing setting, and the actual challenges of the indoor parking problem. Further, we examine the advances made in transfer learning research that equips it to be applied as a new yet effective tool in crowdsensing systems. Finally, we review the recent progress made towards indoor localization and mapping for human users, robots, and vehicles, and also look at various graph matching methods suitable for map-based localization.

2.1 Parking Occupancy Gauging

ParkNet [6] initiates the idea of crowdsensing-assisted parking availability detection, but its reliance on extra sensors (i.e., ultrasonic range finders) installed on ParkNet vehicles has tagged it with a more infrastructure-dependent nature. Bogoslavskyi et al. [17] applies a Markov Decision Process to predict the outdoor occupancy and compute a route that minimizes the **time-to-park** and time to walk to destination. The EU project V-Charge [18] has made advances towards automated parking space detection and navigation, i.e., automated valet parking. Similar to ParkNet, these works rely on onboard cameras, laser or ultrasonic sensors, constraining their scalability for mobile crowdsensing.

Instead of relying on infrastructure, a few of the recent parking systems, ParkSense [11], PocketParker [12], ParkUs [19, 20] and Park Here! [21], take advantage of the driver-vehicle crowdsensing ability: they fuse localization methods and mobile sensing, to detect parking and unparking events and to gauge the availability of outdoor parking spaces. ParkSense [11] relies on the ubiquitous presence of Wi-Fi beacons in urban areas to detect unparking events, while the detection procedure is initiated by a phone-based parking payment system and keeps working until an unparking event is detected. Focusing on outdoor parking areas (surface lot), PocketParker [12] requires GPS or Wi-Fi to roughly locate a parking vehicle and further applies a probabilistic inference mechanism to account for non-participating vehicles. It utilizes the smartphone accelerometer to deduce transitions between driving and walking and thus to detect parking and unparking events [9]. Whereas ParkUs [19, 20] uses inertial sensors and magnetometer instead of Wi-Fi and offers a parking availability heatmap generation function, their core approach is similar to ParkSense and is designed for outdoor parking. Park Here! [21] uses the connectivity status of driver’s personal Bluetooth device with the vehicle’s infotainment system, which is not generalizable.

Though relying on GPS, Wi-Fi, or a payment system is plausible for outdoor parking, it is often not feasible for indoor parking: while both GPS and Wi-Fi are often not accessible, a payment in garage typically happens at the departure time. Also, ParkSense [11] requires the system to keep Wi-Fi sensing active between a pair of parking and unparking events, potentially increasing the overall energy consumption of the smartphone. Even though the use of onboard sensing infrastructure of a vehicle can yield more accurate sensor information, it is not a very scalable approach to obtain sufficient coverage of several parking garages across a large urban area as we will require almost all vehicles to be equipped with such sensors and use the same application. Moreover, inferring occupancy from a subset of parked vehicles (as suggested by PocketParker [12]) may not work well in garages, simply due to their much larger capacity than outdoor parking areas.

Occupancy inference solutions have certainly been proposed to other problems, such as indoor pedestrian flock (cluster) detection [22], railway car-level congestion/occupancy estimation [23], and vehicle on-street occupancy monitoring [24]. However, these methods also suffer from the need for high user participation rates, availability of Wi-Fi

or Bluetooth connectivity from user devices, and/or RF infrastructure, making them impractical in our context.

Crowdsourcing applications are available for drivers to report the occupancy information (e.g., Google Open Spot [25]). Such approaches are often susceptible to failure when they lack proper incentives to stimulate driver participations [11], because a driver would deem it too troublesome to input data manually. In addition, we note that existing vehicle navigation systems provide a useful feature called *estimated time of arrival* (ETA) at destination. However, if a parking system only delivers the number of available parking lots, a driver has insufficient information at his disposal to plan the arrival at the final destination.

2.2 Mobile Crowdsensing

With the proliferation of sensor-equipped mobile computing devices, and smartphones in particular, mobile crowdsensing has been actively researched and deployed for countless applications and a wide variety of domains. This includes medical and healthcare, intelligent transportation, load and occupancy monitoring, resource-sharing services (e.g. car/bike sharing), citizen welfare and reporting, electronic/mobile-governance, search and rescue operations, environmental hazards and pollution monitoring, enhanced social networking, online product advertising and marketing, search recommendations and so on. Intelligent traffic monitoring and prediction systems [26, 27] have been one of the most important and popular application domains of mobile crowdsensing systems, especially because traffic congestion is a global issue that affects most of the urban population around the world.

Both *participatory* sensing [10] which requires intensive user involvement, as well as *opportunistic* sensing [8, 10] which is largely autonomous, have been explored widely. However, many of the participatory sensing applications (e.g. Google Open Spot [25]) their core strength and weakness - human user participation and the lack of it in the absence of strong motivating factors. Thus, most of the successful projects including many works addressing outdoor parking occupancy inference problems [9, 28] as well as outdoor/indoor pedestrian localization [29, 30], rely on opportunistic techniques with only minimal involvement from their users. Therefore in order to gauge parking occupancy or to localize a vehicle, the focus must be on opportunistic crowdsensing of the

parking characteristics or the spatio-temporal location features of the parking garage, in a mostly *autonomous* (or automated) manner. This implies that in such systems, the user’s involvement and responsibility must ideally be limited to ensuring that the smartphone is alive, powered on and connected to data communication networks (if need be), and optionally, providing a destination as input.

2.3 Context and Activity Recognition

Existing literature on human activity and transportation mode detection is bountiful [13, 26, 27, 31, 32, 33, 34]. These proposals often have a low-power profile due to the use of inertial sensing techniques such as [34]. Previous studies [35] have likewise explored the feasibility of using hierarchical Bayesian non-parametric methods for mobile context discovery from raw sensor data.

However, it is difficult to directly apply inertial sensing (which simply indicates motion) to derive the parking occupancy. Fortunately, recent advances in machine learning and data mining techniques allow indirect inference to be performed if proper logical correlations can be identified [36]. Therefore, it is potentially feasible to avoid the sensing-prohibitive direct counting of parking lots and still gauge the occupancy from seemingly irrelevant sensor readings.

2.4 Transfer Learning for Regression Tasks

Bootstrapping large scale crowdsensing system deployments under unseen target environments (such as new parking garages) is challenging when training data is insufficient. Although self-sustaining deployments that subsist by consuming data contributed by own users are plausible, it is often considered a chicken-or-egg issue. Transfer learning methods can leverage on source data or models from existing deployments, spanning diverse cases where the source and target data may belong to different distributions or even dissimilar domains [37]. In such cases, the traditional machine learning methods requiring a consistent distribution of data may not perform well.

Fortunately, we have instance-transfer methods such as TrAdaBoost.R2 [38] and its two-stage variant utilizing cross-validated AdaBoost.R2 models [39, 40] to solve regression transfer problems such as parking occupancy inference, by applying the

boosting principle. The latter offers robust performance against overfitting due to the cross-validation steps and is hence promising to transfer the concepts of parking occupancy inference. Nevertheless, how to apply transfer learning methods to improve the scalability of mobile crowdsensing systems is still an open question. This warrants a novel direction to the mobile computing research community.

2.5 Mobile Sensing for Indoor Localization and Mapping

Mobile indoor localization and tracking efforts mostly utilize location fingerprints based on wireless Received Signal Strength (RSS) [41, 42], primarily from WiFi Access Points (AP) and also from Bluetooth Low Energy (BLE) beacons or cellular radio. Since RSS-based methods can be quite energy-hungry, alternative approaches have been explored for indoor pedestrian localization and mapping by combining inertial sensing with other data sources such as geo-magnetism and landmark imagery. For example, UnLoc [43] detects organic landmarks and its extension SemanticSLAM [44] applies FastSLAM [45] for unsupervised learning, and Jigsaw [46] applies photo-based indoor floor plan construction. Nawaz et al. [47] have demonstrated the usage of smartphone gyroscope data for GPS-free macro-scale localization by learning a driver’s frequently traveled routes from the sequence and characteristics of turn maneuvers.

These methods are not directly applicable to vehicle localization, as they rely on human step-counting for odometry and use WiFi or geo-magnetism for landmark recognition. Unfortunately, we do not have any known equivalent *periodic* pattern that can be sensed during the smooth vehicular motion in an infrastructure-free manner, using only a mobile phone. In contrast, traditional car navigation systems get odometry information by counting wheel revolutions using proprietary vehicle sensors that are not accessible¹ to common users. WiFi is usually unavailable in underground garages, and prohibitive interference by in-vehicle magnetic fields render geo-magnetic fingerprints

¹Although the in-vehicle sensor information is partially reported through the OBD-II port available on almost all vehicles, it is very limited and is mostly insufficient for localization. On the other hand, a few vehicle manufacturers have opened up some additional proprietary data on specific vehicles using special access mechanisms and/or software API (for e.g., the OpenXC platform from Ford) but this is still restricted to a few vehicle models and there is no common standard yet. With a more widespread adoption of “connected car” methodologies (including 801.11p and other V2X standards) in future, we can expect vehicles to distribute more information; nevertheless, these are currently under experimental phase and may take many more years for commercial deployment and public usability in general.

ineffective during parking. However, their energy-efficient methods offer a potential to be merged into ParkLoc, to deliver a seamless navigational experience to drivers walking between their destination and the parked vehicle.

2.6 Vehicular Localization and Mapping

Traditional vehicular localization approaches use filtering approaches such as the popular Kalman Filter, its Extended version (EKF), and the Particle Filter [48, 49, 50]. However, most recent works in robotics tend to use Simultaneous Localization and Mapping (SLAM) [48]. SLAM offers two distinct paradigms: *metric-based* and *topological* [51]. To navigate a human-driven vehicle in constrained environments such as indoor parking garages, a globally consistent metric map is an overkill; instead, a map with rough topological connections between maneuver points and/or landmarks is sufficient - if their global positions are known or determinable.

Although graph-based approaches [52] such as GraphSLAM [48, 53, 54] have become increasingly popular in robotics in the last two decades, they have not received comparable attention in mobile computing systems where *filtering* approaches, mostly particle filter, seem to be the tool of choice. This is the case even for mobile vehicular localization, where the paths navigable by a vehicle possess a planar graph structure, whether on outdoor road networks or within indoor parking garages [55]. Only a handful of mobile systems [56, 57, 58] have explored this potential, mainly utilizing WiFi RSS. A key reason for this research gap could be the mismatch in sensing capabilities and training data quality between robotic systems and handheld mobile devices. However, with some careful considerations of the parking garage environment, we can still take advantage of some virtual yet intrinsic landmarks defined by their spatio-temporal constraints, and incorporate them into GraphSLAM.

CarLoc [59] applies instrumentation to obtain vehicle sensor data to make odometry/orientation estimates and combines them with roadway landmark information derived through computer vision techniques into a particle filter to improve localization accuracy. VeLoc [15] and VeTrack [16] propose a smartphone-only particle filter-based framework with a road skeleton model representing garage map constraints and a supervised turn/bump detection method to track the vehicle within indoor parking facilities. Whereas CarLoc relies on GPS and proprietary vehicle sensors and is designed for

outdoor, VeLoc/VeTrack requires a garage map and large training data collection and training per garage. Since it is difficult for smartphone applications to get instrumented access to proprietary vehicle sensors and because high training requirements affect the scalability of deployments, it is important to develop alternative light-weight indoor vehicle localization approaches easy to deploy and operate.

2.7 Graph Matching

The pattern recognition research community has made several efforts, over the past several decades, to address the problem of matching two similar graphs, both in an exact (perfect) and an inexact (approximate) manner. The seminal work by Ullmann [60] provides the classic subgraph isomorphism method that has been widely used over the last 3 decades. Nauty Algorithm [61] and VF2 [62] are some of the other popular algorithms to solve this problem more efficiently. These methods have been adopted in various applications and disciplines. However, to the best of our knowledge, their potential is yet to be fully utilized in graph-based localization systems and mobile computing research.

2.8 Summary

In this chapter, we have investigated the efforts made by mobile computing community to address the twin problems of parking occupancy inference as well as vehicular localization. We also review the advances made on human activity or mobility context recognition, crowdsensing and transfer learning. We realize that most of the existing solutions rely on infrastructural supports for precise localization which are typically unavailable at indoor parking garages. We also note the practical issues faced by mobile crowdsensing system deployments with regard to the scope for user involvement or the need for large training data collection and labeling. Therefore, in this thesis, we attempt to make a few pragmatic innovations in the form of methods for scalable parking occupancy inference, and light-weight indoor vehicular localization and mapping, with a long-term vision of pushing forward the large-scale adoption of ubiquitous and potent mobile sensor data that can empower a host of novel intelligent mobility applications and services.

Chapter 3

ParkGauge+: Towards Scalable Occupancy Inference for Parking Garages

In this chapter, we closely examine the issue of gauging the occupancy of indoor parking garages using only commodity smartphones, by using a prototype system named ParkGauge+. In a nutshell, we present ParkGauge+ as a mobile sensing system that does not require any infrastructural support including GPS and/or WiFi, except to kick start the sensing scheme upon arrival. Unlike the existing proposals that mostly count the parking spaces based on parking/unparking events, we innovate in the use of a data-driven crowdsensing approach that leverages temporal parking characteristics such as *time-to-park* to train models to gauge the parking occupancy. Furthermore, we propose a transfer-learning approach to scale the system to new garages without major data collection effort. In this chapter, we motivate the need for ParkGauge+, propose the architecture and detailed methodology, and finally our extensive and realistic experimental evaluation results.

3.1 Introduction

The vehicle ownership in dense urban areas has been witnessing extreme growth around the world, fuelled by rapid economic growth; this has led to a higher demand on parking

Figure 3.1: Parking garages in the CBD area of Singapore, with percentages of the types of occupancy monitoring shown on the left. Garages with online occupancy information and on-street parking are marked in green and purple, respectively.

infrastructure especially in mega-cities and in developing countries. It is well-known that on-street parking produces serious traffic congestion due to vehicles cruising to find parking places [2, 63]. On the other hand, *multi-storey car parks* (aka, *parking garages*), which are the major urban parking facilities of mega-cities, can also generate serious on-road congestion because of long queues of vehicles waiting to enter them, even though their total capacity may be sufficient to meet the parking demands. A reason for this paradox is that the occupancy information measured at the garages (if any) is mostly displayed only locally, rather than published online and made available to drivers in advance through roadside displays or smartphone apps. As illustrated in Figure 3.1, the data from our study on the central business district (CBD) of Singapore during the year 2017 shows that, out of 360 garages identified, 22.5% have no occupancy monitoring system installed and 92% do not publish the real-time occupancy information online - this can cause waiting queues as shown in Fig. 3.2.

Vehicles that are searching or waiting to get parked also cause environmental issues, as their emissions produced during cruising/braking result in serious air pollution [6, 63]. Many cities try to control the problem by establishing road-side parking guidance systems, adding occupancy sensor infrastructure, and even enforcing special policies (e.g., a dynamic parking pricing scheme) to discourage people from driving into a CBD,

Figure 3.2: Queuing at indoor parking garages: (top) at entrance and (bottom) the interior

but they all need heavy investment to be effective and some may actually increase the overall carbon footprint. In fact, the majority part of the parking problem can be tackled if the occupancy information is available to drivers in advance, enabling them to drive towards less congested venues or re-plan their trips. Our studies reveal that some popular parking garages may remain full for several hours with acute queuing even when close-by ones possess surplus, as exemplified by the three garages A, B and C shown in Fig. 3.3, whereas drivers may be oblivious of this reality. Thus, it would be extremely useful if we can have an intelligent system that helps the driver to answer a simple but crucial question¹: *Can I park when I arrive?*

Existing infrastructure-based systems deployed to collect parking occupancy data often require occupancy sensors and wireless beacons at parking lots (e.g., SFpark [5]) or extra sensors on vehicles (e.g., ParkNet [6]), making them expensive and not very scalable. For small garages, an ingress-egress detection system installed at the gates may seem to be sufficient to monitor the occupancy. However, this is often inadequate in larger multi-storey garages, because even after gaining entry, the vehicles may not

¹As a by-product, such a pragmatic intelligent system can also help the driver to answer the question: *When can I complete parking and leave my car to arrive at my actual destination?*

Figure 3.3: Parking availabilities for three close-by parking garages A, B and C during a day: even when A and C are both fully occupied (0 available spaces), B has plenty!

immediately find a space convenient to the driver’s preferences, leading to indoor congestion (as shown in Fig. 3.2) and uncomfortably long **time-to-park**. To circumvent this issue, garage operators could install sensor infrastructure to monitor each space. However, the operators of parking garages at many *emerging* economies with rising vehicle ownership are unable to deploy infrastructure-based systems due to their prohibitive costs, given the low parking revenue obtainable in comparison to their counterparts in affluent cities. Consequently, it would be very helpful, from both economic and environmental points of view, if the crowdsensing ability [8, 10] of the driving population can be exploited to tackle the parking problem in an infrastructure-free manner.

ParkSense [11] and PocketParker[12] have recently demonstrated the crowdsensed detection of on-street/outdoor parking occupancy, by taking support from infrastructure-based precise localization schemes such as WiFi and/or GPS which are mostly unavailable to indoor parking garages. Inertial sensing has been widely used for human activity and transportation mode detection (e.g., [13, 27, 31, 33, 64]) without any infrastructure support, but it is extremely hard to infer occupancy information from direct inertial sensor readings: inertial sensing may well suggest driving states such as **braking** and **turning**, but such events seem unrelated to parking occupancy - unless we place inference mechanisms to establish the necessary connections. Deploying a useful crowdsensing system typically involves substantial data collection. In the ideal situation, a parking occupancy crowdsensing system can be trained and calibrated for each individual parking garage, as suggested by most of the existing proposals. However, this is hard to achieve in practice, due to the costs and efforts involved. This leads us to a general

question: *Can we scale a crowdsensing system to additional deployments sans major data acquisition?*

In this chapter, we propose ParkGauge+, a mobile crowdsensing system to gauge the occupancies of parking garages in urban areas. To apply inertial sensing for this purpose, ParkGauge+ innovates in using (temporal sequences of) driving states (e.g., braking, turning and pausing) to infer parking characteristics (e.g., time-in-queuing and time-in-cruising) that are in turn exploited to deduce garage occupancies. To achieve scalability, we propose to employ *inductive transfer learning* techniques [37] by combining the training data from several parking garages to gauge the occupancy of a new garage with only minimal training data. To the best of our knowledge, ParkGauge+ is the first *transfer crowdsensing* method that uses transfer learning to learn from crowd-sensed temporal characteristics of multiple indoor parking garages and indicate the occupancy. The primary contributions of this work are summarized as follows:

- We propose a fundamentally new idea to measure the occupancy of parking garages, exploiting seemingly irrelevant yet ubiquitous mobile sensor data.
- We develop a full sensing architecture in order to complete the transformation from (mostly) inertial sensor readings to temporal parking characteristics.
- We identify, combine and re-engineer machine learning and data mining algorithms in a hierarchical manner, aiming to guarantee a highly effective inference process on the diverse datasets typically obtained through a crowdsensing system.
- We apply inductive transfer learning to the parking occupancy inference problem, by combining data from different parking garages, resulting in a scalable approach that minimizes training efforts and costs.
- We demonstrate the efficiency and effectiveness of ParkGauge+ with extensive experiments in several parking garages, using a prototype smartphone application implemented on Android platform.

It is worth noting that, unlike existing crowdsensing-based parking systems that directly count the available parking lots, ParkGauge+ does not require a “crowd” (hence high penetration of the application) for sensing individual parking garages. Instead, a minimal amount of sensing data acquired from a small number of users for each parking

Figure 3.4: Illustration of temporal parking characteristics and driving contexts of a vehicle at an indoor parking garage.

garage would be sufficient for ParkGauge+ to deliver useful information, whereas the “crowd” is needed only for covering many parking garages across a large urban area. However, even such a requirement for a large spatial distribution of a “crowd” can be minimized by reusing source datasets from existing deployments, transferring and learning from them to re-train ParkGauge+ for new target garages.

3.2 Overview of ParkGauge+

This section explains the design rationale of ParkGauge+, followed by an overview of the architecture.

3.2.1 Design Rationale: Why Temporal Characteristics?

Whereas directly counting the available parking lots as has been suggested in the current proposals is not feasible for indoor parking garages, it is not effective either because the crowdsensing application has to be adopted by most (if not all) drivers, which is difficult to achieve in practice. Therefore, ParkGauge+ aims to gauge the occupancy evaluation from a different perspective. In particular, if we deem a parking garage as a queueing system, the occupancy determines its service rate, which is functionally related to the waiting time in the queue. In the parking context, we term this waiting time *time-to-park* and consider it as a temporal *parking characteristic*; it is composed of other parking characteristics such as *time-in-cruising* and *time-in-queuing* as illustrated in Fig. 3.4. While *time-to-park* is defined as the time span from a vehicle’s arrival at

Figure 3.5: Parking occupancy and time-to-park at a popular shopping mall: a negative correlation is obvious.

a parking garage till it gets parked,¹ *time-in-cruising* and *time-in-queuing* measure the time spent in looking for a free parking lot and that for waiting in a stationary state,² respectively. Fig. 3.5 illustrates the strong negative correlation between parking occupancy and *time-to-park* at a popular shopping mall, which in this particular example can be quantitatively represented using Pearson correlation coefficient of -0.89.

Using parking characteristics such as *time-to-park* to infer occupancy can also satisfy our other design objectives. First of all, these characteristics can be detected mostly by inertial sensing techniques (if properly used), making the system both autonomous and energy-efficient. Secondly, it is scalable to very large metropolitan areas: since only a few tens of samples (so as to reach certain statistic significance) need to be obtained for each parking garage, a small number of ParkGauge+ equipped drivers/vehicles would be able to cover a very large area. Thirdly, the data used for inferring occupancy can extend to serve other purposes, including a more accurate ETA indication. Finally, the system is self-incentivizing: if drivers wish to get the occupancy information for their respective destinations, they would enable the ParkGauge+ application to share their data in the first place.

3.2.2 System Architecture

ParkGauge+ primarily runs as an application in a driver’s smartphone that is assumed to be taken along with the driver upon completing a parking. The architecture of

¹We do not define the starting time as when a vehicle enters a garage, since waiting in a queue can also happen outside the garage, as shown in Fig. 3.2.

²According to our experience, vehicle queues may exist both at the entrance of a parking garage and inside of it.

Figure 3.6: Architecture of ParkGauge+.

ParkGauge+ has a hierarchical presentation shown in Fig. 3.6. There are essentially 2 major components: the *ParkGauge+ Sensing Core*, that computes parking characteristics from seemingly unrelated raw sensor data, and the *Scalable Parking Occupancy Estimator* that uses the parking characteristics dataset to gauge the parking occupancy.

ParkGauge+ collects raw data using several sensors embedded in the smartphone, mainly gyroscope, accelerometer, and barometer for continuous sensing in a parking garage, as well as location sensors (3G, GPS, or Wi-Fi) only to determine the starting context of a parking process. Within the Driving State Detector, features extracted from the sensor data collected in a sliding window are fed into a Random Forest-based classifier that estimates the *driving state*, such as **accelerating**, **braking** and **turning**. The Driving Context Detector chiefly applies a *Hidden Markov Model* (HMM) to represent the relation between the input (*driving state*) and the output (*driving context*), where *driving states* are treated as observable states and the parking-relevant *driving contexts*

are the hidden ones to be inferred. As the HMM factors in the temporal correlations between consecutive driving contexts, it potentially counteracts the errors in classifying the driving states. Note that a few contexts such as **start/give-up parking** or **floor-change** require direct sensor inputs, resulting in connections that bypass the HMM.

Parking characteristics are derived from *driving contexts*. For example, the time span between **start-parking** and **parked** is counted as **time-to-park**, whereas **brake count** counts the number of brakings. Some characteristics can be used directly (e.g., to improve ETA estimations). However, inferring the occupancy of a parking garage requires an independent learning module based on regression. Using the historical data, we build a regression model to represent the relation between occupancy and the parking characteristics, so that ParkGauge+ can infer the occupancy in real-time.

For parking garages with sufficient “local” training data of high quality (as attested by certain performance metrics), the Scalable Occupancy Estimator employs Support Vector Regression (SVR) with a universal kernel function based on Pearson VII function (PUK) [65]. However, in case of garages with low-quality data, ParkGauge+ instead activates the Transfer Learning for Regression (task) to exploit the richer “external” parking characteristics data from other garages where ParkGauge+ is already deployed. A boosting-based approach, namely, *two-stage TrAdaBoost.R2* helps to identify the useful data instances and learn the best regression model to predict the parking occupancy. As all the learning models are trained offline, the online inferring process run by ParkGauge+ client involves only low complexity arithmetic operations, making it both time and energy efficient to be hosted in commercially-available smartphones.

Table 3.1: List of states, contexts and parking characteristics.

Driving state	Driving context	Parking Characteristic
a. proceeding	i. driving	1. time-to-park
b. accelerating	ii. cruising	2. time-in-queuing
c. decelerating	iii. parked	3. time-in-cruising
d. turning	iv. queuing	4. day of week
e. braking	v. start-parking	5. arrival time
f. pausing	vi. give-up-parking	6. brake count
g. walking	vii. floor-change	7. turn count

Table 3.1 lists all the driving states, driving contexts and parking characteristics supported by ParkGauge+.

3.3 Methodology

We explain the key aspects of our methods in this section. We first focus on scalable occupancy inference with known parking characteristics, then we explain how to acquire these characteristics via mobile crowdsensing.

3.3.1 Scalable Parking Occupancy Inference

Our occupancy inference method is logically separated into local inference and transfer inference, with the latter to improve the scalability of ParkGauge+.

3.3.1.1 Inference with only Local Data

As explained in Section 3.2.1, there exists an implicit functional relationship between **time-to-park** and occupancy. Instead of deriving it analytically, we use regression to fit this function to our labeled "local data" (*c.f.* Section 3.4.4.1) from a parking garage. As a result, whenever ParkGauge+ obtains an estimated **time-to-park**, it can gauge the corresponding occupancy. Such occupancy estimations can be averaged over a sliding time interval (for example, 20 minutes duration in our reference implementation) and reported to users in real-time. Figure 3.7 illustrates this relation using real-world data collected from a popular parking garage during a working day (10:00 to 16:00) with at least 1 observation recorded every 10 minutes. It can be observed that lower **time-to-park** is observed during periods of high parking availability and vice-versa. When such high-quality data are available, a simple linear or quadratic regression fits a model that can operate with reasonably low estimation errors.

To further improve the inference accuracy, we propose to use additional parking characteristics computed by the ParkGauge+ core for each parking session and use them offline to train a regression model. A robust regression model can be obtained using Support Vector Regression (SVR) with an appropriate kernel function selected through experimental evaluation (see Section 3.4.4.1).

Figure 3.7: A basic regression model that can fit and recover the relation between parking occupancy and time-to-park. The red dashed line indicates a simple linear fit, whereas the blue solid line indicates a quadratic fit.

Support Vector Regression

There has been extensive literature on Support Vector Regression [66] and its several applications in diverse research areas including intelligent transport systems, such as for travel-time prediction [67]. Hence we only briefly describe the relevant concepts here. Given a training data set of n observations $F = (x_i, y_i), i = 1, 2, \dots, n$ with $x \in \mathbb{R}^d$ representing the d input features (parking characteristics) obtained from each parking session and $y \in \mathbb{R}$ representing the corresponding output parking occupancy to be inferred, SVR tries to find the function $f(x): \mathbb{R}^d \rightarrow \mathbb{R}$ mapping the input training data features x to output y , that produces estimation of y which is the closest to the corresponding true values in F . The problem belongs to Reproducing Kernel Hilbert Spaces (RKHS) $\mathcal{H}$ and can be formulated as:

$$f^*(x) = \arg \min_{f \in \mathcal{H}} \left(C \sum_{i=1}^l V(f(x_i), y_i) + 0.5 \|f\|_{\mathcal{H}}^2 \right) \quad (3.1)$$

where C is a parameter controlling the relative weight of the loss function and the regularization penalty $\|f\|_{\mathcal{H}}^2$. The loss function in our case is $V(f(x), y) = |\text{round}(f(x)) - y|$.

Choosing the Kernel Function

Kernel-based methods have become highly popular in solving regression and classification problems and several kernel functions have been proposed. Kernel functions help to transform the input data space into a Hilbert Space (i.e. a higher-dimensional space

which is spanned by distance-based functions or inner-products of the input feature data vectors $\in \mathbb{R}^d$). Some of the popular kernel functions are Linear kernel, Polynomial kernel and Radial basis function (RBF). However, most of these functions are either too trivial to provide enough flexibility or too complicated (e.g., with many parameters) to fine-tune.

Recently, [65] proposed the Pearson VII Universal kernel (PUK) function i.e. the Pearson VII function developed by Karl Pearson in 1895 [68]. PUK can be regarded as a universal kernel since it can handle linear, polynomial and RBF based feature mappings. It possesses the flexibility to vary between Gaussian and Lorentzian shapes and more, thereby offering more robust mapping capability and enabling it to deal with a broad spectrum of data and problems. Since PUK can help avoid the tedious process of selecting kernel functions, the model building process can be made simpler and computationally efficient. The Pearson VII function can be understood in the general form

$$\frac{H}{\left[1 + \left(\frac{2(x-x_0)\sqrt{2^{(1/\omega)}-1}}{\sigma}\right)^2\right]^\omega} \quad (3.2)$$

where x is the feature vector and H represents the maximum height of the peak of x (i.e., the value of y observed at its center x_0). The half-width and tailing factor of the peak are controlled by the parameters σ and ω . The function offers the flexibility to change its shape from Gaussian ($\omega \approx \infty$) towards a Lorentzian ($\omega \approx 1$) shape and other intermediate shapes by varying the parameter ω . This property enables PUK to fit peaks of a variety of line widths and shapes, thus helping to serve as a universal kernel function.

3.3.1.2 Scalable Inference with Transfer Learning

In this section, we describe how can ParkGauge+ be scaled up to gauge the occupancy of new parking garages with insufficient “local” training data, and explain a boosting-based algorithm for regression transfer learning.

Why Transfer Learning?

The procedure described in Section 3.3.1.1 helps us to gauge the occupancy for a single parking garage, for which we possess a good amount of “local” training data with

sufficient quality. This may well suggest that sensor data and ground truth have to be acquired afresh at each garage where ParkGauge+ is to be additionally deployed. Such data collection efforts can be expensive and in some cases, even increase the time taken to deploy ParkGauge+. Fortunately, we have the possibility to opportunistically exploit the datasets and knowledge available from existing ParkGauge+ installations and thereby facilitate a faster deployment of ParkGauge+ at new target garages. To this end, we propose to use *inductive transfer learning* methods.

In general, *transfer learning* deals with the problem of learning a new target concept using a small amount of target data, and given the knowledge of related source concept(s) supported by large source datasets (that we refer to as "external" data). Whereas the method in Section 3.3.1.1 uses classical machine learning techniques under the assumption that the domain and distribution of training and testing datasets are the same (e.g., from the same parking garage), transfer learning has to handle cases where they are indeed different! [37] To enable a scalable performance in such situations, ParkGauge+ supports learning from "external" parking garage datasets with differences in structure, usage patterns, rules etc.

Boosting-based Transfer Learning for Regression The inductive transfer learning is applicable when the source and target tasks are always different. In the inductive *instance-transfer* approach, we process the entire source dataset (from well-trained ParkGauge+ deployments) but select only the "relevant" source data instances that possess a strong predictive ability to gauge the occupancy of the target parking garage. To solve regression problems in the transfer learning context, we apply the *TrAdaBoost.R2* algorithm [38], which uses the classic *AdaBoost.R2* ([39], [40]) to boost only the target data instances. The weakly performing source instances (which are the least correlated with the target distribution and lacking sufficient predictive ability) are rejected, whereas good performing instances are retained. Specifically, we propose to apply a modified *two-stage TrAdaBoost.R2* algorithm [38] that applies the transfer boosting principle to cross-validated AdaBoost.R2 models. It potentially offers better generalization performance and can be more robust against overfitting.

To perform regression transfer, we consider two labeled training data sets: D_S (of size n), a rich source dataset from garages where ParkGauge+ was trained and found to produce high performance, and D_T (of size m), a small target dataset for the garage

Algorithm 1: Modified *Two-stage TrAdaBoost.R2*[38]

Input: Two labeled training sets D_S (size n) and D_T (size m), S the number of steps, N the maximal number of boosting iterations, F the number of cross-validation folds, $bLrn$ the base regression learning algorithm, $\mathbb{R}$ the set of real numbers

Output: $model_t$ where $t = \argmin_i error_i$

- 1 Let $D = D_S \cup D_T$ s.t. $D_{i=1}^n \in D_S$, $D_{i=n+1}^{n+m} \in D_T$. $D = \{(x_i, y_i)\}_{i=1}^{n+m}$ where $x_i \in X$ the space of parking characteristics reported by users and $y_i \in Y$ the space of corresponding true occupancy values
- 2 Initial weight $\mathbf{w}^1$ s.t. $w_i^1 = 1/(n+m)$ for $1 \leq i \leq (n+m)$
- 3 **for** $t = 1, \dots, S$: **do**
 - 4 1. Run *AdaBoost.R2*: get $model_t$ from D with sampling distribution $\omega^1 = \mathbf{w}^t$ using $bLrn$, $N^t = N$
 - 5 **for** fold $D_f \subset D$ s.t. $f = 1, \dots, F$; $k = 1, \dots, N^t$ **do**
 - 6 a. Use $bLrn$ to get a hypothesis $h_k : X \rightarrow \mathbb{R}$
 - 7 b. Calculate adjusted error e_j^k per instance:
 - 8 let $e_{max}^k = \max_{j=1}^l |y_j - h_k(x_j)|$, $l = \text{len}(D_f)$
 - 9 then $e_j^k = |y_j - h_k(x_j)| / e_{max}^k$
 - 10 c. Calculate adjusted loss of h_k , $\epsilon_k = \sum_{j=1}^l e_j^k \omega_j^k$
 - 11 if $\epsilon_k \geq 0.5$, stop and set $N^t = k - 1$
 - 12 d. Update the weight vector iff $D_j^k \in D_T$
 - 13 $\omega_j^{k+1} = \omega_j^k \beta_k^{1-e_j^k} / Z_k$ s.t. $\beta_k = \epsilon_k / (1 - \epsilon_k)$
 - 14 and Z_k is a normalizing constant
 - 15 **end**
 - 16 Get $model_t = \text{weighted median}(h_{k=1}^{N^t}, \ln(1/\beta_k))$
 - 17 Use F -fold cross validation to obtain an estimate $error_t$ of the error of $model_t$
 - 18 2. Call $bLrn$ with D and distribution $\mathbf{w}^t$, and get a parking occupancy hypothesis $H_t : X \rightarrow \mathbb{R}$
 - 19 3. Compute the adjusted linear loss function E_i^t for each instance i as:
 - 20 where $E_{max}^t = \max_{i=1}^{(n+m)} |y_i - H_t(x_i)|$
 - 21 4. Update the weights $\mathbf{w}^{t+1}$ for next step:
$$w_i^{t+1} = \begin{cases} w_i^t B_t^{E_i^t} / Z_t, & 1 \leq i \leq (n) \\ w_i^t / Z_t, & n+1 \leq i \leq (n+m) \end{cases} \quad \text{where } Z_t \text{ is a normalizing constant, and } B_t \text{ is chosen such that the resulting weight of the target (final } m) \text{ instances is } \frac{m}{(n+m)} + \frac{t}{(S-1)}(1 - \frac{m}{(n+m)})$$
 - 22
 - 23 **end**

that we intend to learn the regression model.¹ Let S be the number of steps in the second stage, N the maximum number of boosting iterations, and F the number of cross-validation folds. Let $bLrn$ be the base learning algorithm, which can be a simple and fast performing regression algorithm with good accuracy such as least squares linear regression or regression tree. Let D be the full training dataset, that combines D_S and D_T . The first stage of the two-stage TrAdaBoost.R2 algorithm produces an F -fold cross-validated model $model_t$ using AdaBoost.R2 for up to N iterations. This boosts the weights of only target instances based on a linear error (loss) function evaluated by $bLrn$. In the second stage, an adjusted error is computed to update (i.e. selectively boost or dampen) the weights of only source instances. Finally, after convergence or exhausting S stage-2 iterations, the model producing minimum loss is selected to gauge the occupancy of the target parking garage. Algorithm 1 outlines the important steps required for transfer learning of parking occupancy inference.

While this algorithm can generally provide good transfer learning performance, in practice the best results may be obtained while transferring between two garages of similar structure and usage patterns. For e.g., the source and target garages considered could be those are attached to shopping malls and having a similar capacity as well a consistent distribution of floors under or above ground. On the other hand, the performance may suffer if and when transferring between a large but single-storey garage with huge capacity and a multi-storey garage with small capacity per storey, even if their total capacity is the same. This is because in the latter case of multi-storey garages, the top floor may be less attractive due to the number of spiral ramps to climb up and down (that increases both **time-to-park** during arrival and **time-to-exit** while departing). Therefore, key considerations must be made to compare these properties before adopting transfer learning. Another important consideration is the choice of base learner $bLrn$; as discussed above, a good method has to be chosen which offers good accuracy as well as high temporal efficiency of learning.

3.3.2 Sensing Core of ParkGauge+

In this section, we first present two direct-sensing components, namely parking-arrival and floor-change detectors. Subsequently, we focus on the parking process model,

¹The minimal data requirements are discussed under Section 3.5.1

driving state detection and explain how to learn the HMM model from observed sensor data and infer the driving contexts and compute the parking characteristics.

3.3.2.1 Parking-Arrival Detector (PAD)

The contexts **start-parking** and **give-up-parking** need to be captured to set and reset an inference process. ParkGauge+ relies on a low-energy 3G-based location detector to coarsely indicate if the geo-fence around a parking garage near the destination is entered. If true, ParkGauge+ starts the GPS-based localization to capture either **start-parking** or **give-up-parking** context. If the vehicle further approaches the garage close to the entry gate, the **start-parking** context is signaled and ParkGauge+ starts its inference process. The current time is recorded as the candidate *Parking Arrival Time*, and the Driving Context Detector (DCD) is invoked. Under a normal situation, the false-positive rate of detecting a **start-parking** is low, as the chance of a vehicle slowly approaching a parking garage but bearing no intention of entry is very small. However, vehicles driving near a parking garage but hit by a traffic jam can also cause a false alarm. Fortunately, ParkGauge+ is robust to such anomalies, as the recorded *Parking Arrival Time* is only a candidate; it can be canceled if a **give-up-parking** is signaled, either due to the vehicle exiting the geo-fence or as a result of the driving context signaled by DCD (*c.f.* Sec. 3.3.2.3).

3.3.2.2 Floor-Change Detector (FCD)

This detector primarily operates on the barometer sensor data to indicate relative height variations and thereby detects floor changes during a vehicle’s parking process. Detecting **floor-change** context allows ParkGauge+ to reinforce its belief in an ongoing parking process, and it also facilitates deriving parking characteristics for individual floors within a parking garage (e.g., **time-in-cruising** for basement 2), so that ParkGauge+ can provide fine-grained parking characteristics desirable for users who, for example, prefer to park at a floor close to their favorite shops.¹

¹Such profile information can be of interests to other mobile applications such as advertisement pushing or coupon distribution.

Figure 3.8: Barometer atmospheric pressure variations while changing floors (going downwards) in a parking garage with 3 floors underground.

Figure 3.8 depicts the variations in barometer reading after a vehicle enters a parking garage and then goes two floors downwards. This particular garage offers direct access to each floor through a spiral ramp.

While the gradual increase in barometer reading is the consequence of the vehicle’s cruising along a spiral ramp, the overall variations clearly indicates the floor changes: the altitude variation is inversely proportional to that of the pressure with a slope of roughly 0.12 hPa/m. The patterns are consistent for each parking garage and that helps ParkGauge+ offer repeatable floor detection performance even in the presence of long-term pressure fluctuations [69]. ParkGauge+ obtains barometer readings at 1Hz frequency and it applies a 10s window to check if the readings are sufficiently stable, i.e., variance below a threshold (0.1 hPa). Whenever stable readings are identified, their average value is then compared with that of the previous one. If the resulting altitude difference falls within a predetermined height interval¹ applicable for the parking garage structure (e.g., around 3.3m in general, or 4.4m in special cases for ground

¹To ascertain the robustness of this deterministic approach, we examined the floor heights at several commercial parking garages in Singapore and found the typical inter-floor height to be around 3.2-3.3m; the major exception is at the ground floor of some buildings having a higher headroom clearance to allow passage of heavy vehicles and hence a height of 4.3-4.5m between ground floor and the first floor above. Since the parking ramps we evaluated are constructed with a stipulated gradient within 1:20 to 1:25 (as per building construction code), we expect the slope of altitude variation remain the same (within a very small range of variations, up to 1%) although the ramp distance could be slightly different at different floor heights. Given the above observations, we believe that considering these two inter-floor height intervals are sufficient to implement a good performing FCD.

floor wherever relevant), a floor-change context is signaled.

3.3.2.3 Driving Context Detector (DCD)

DCD is a central component of ParkGauge+ that infers the driving contexts of a vehicle. We first briefly introduce the basic models. Then we explain the feature selection and driving state classification at the sensing layer. Finally, we present the learning process for the HMM (context layer).

Figure 3.9: Model of a parking process used by DCD.

Modeling a Parking Process

Here, we describe our inference model for the whole parking process, as depicted in Fig. 3.9. We use O and C to represent *observation* (or driving state¹) and driving context respectively. The time-granularity of inference is quantified using a window of size W . We use the subscript t to denote the time index of the state and context for a window. Except for the 3 direct-sensed contexts discussed in Sec. 3.3.2.1 and 3.3.2.2, the relation between observation (driving state) and driving contexts are modeled as an HMM with standard first-order Markov assumptions made for the temporal sequence of driving contexts. This can be specified by: (1) stationary context distribution, $\Pr(C_1)$, (2) context transition model, $\Pr(C_t|C_{t-1})$, and (3) observation model, $\Pr(O_t|C_t)$.

When a stream of driving states $[O^1, O^2, \dots]$ are estimated by the sensing layer, ParkGauge+ takes them for inferring the t -th driving context C_t , where the correlation between consecutive contexts has been implicitly considered by involving the previous context as an input. Table 3.1 lists all the driving contexts and observations considered.

¹We shall use these two terms in an interchangeable manner.

Table 3.2: Features used in classifying driving state.

Acceleration-based		Gyroscope-based
Time-domain	Frequency-Domain	Time-domain
Mean	Peak Frequency	Mean
Variance	FFT-Peak	Variance
Range	FFT-0,1,2,3,4Hz	Range
Gradient	Energy	Peak Count

Feature Extraction and Driving State Detector

As shown in Fig. 3.6, we mainly use the accelerometer and gyroscope for driving context detection. Whereas barometer may be used for this purpose with even lower energy consumption [14], our current implementation avoids using it due to its longer detection latency. Next, we describe how features suitable for ParkGauge+ are extracted and used to train a classifier that estimates driving states.

ParkGauge+ uses a sampling frequency of 20Hz to collect sensor readings through Android SensorManager API. Note that optionally, an axes alignment scheme can be employed to estimate the phone attitude and align the sensor data to the vehicle axes, by following the methods from literature as described in Chapter 4 under Section 4.2.2.2, although it is not necessary if the phone is kept in a stationary position while driving. The driving state is determined by collecting these readings in an interval of W seconds and feeding it to a classifier. By default, $W = 10$ s and we justify this choice through experiments in Sec. 3.4.3.3. Raw acceleration values are pre-processed by applying a low-pass filter to estimate the gravity component that is removed to obtain linear acceleration values. These values are further smoothed (to mitigate noise) by applying a 5-term moving average, and the results are processed to extract the time-domain and frequency-domain features listed in Table 3.2. Similarly, raw gyroscope readings are also smoothed and processed to extract the specific angular time-domain features listed. Fig. 3.10 shows examples of raw sensor readings corresponding to the ParkGauge+ driving states. Clearly, the aforementioned features characterize these states sufficiently.

Using datasets obtained from our experiments (*c.f.* Sec. 3.4.1), we train a Random Forest (RF) based classifier [70] offline that enables ParkGauge+ to efficiently and

accurately estimate the driving state from sensor data at run-time. The training dataset consists of the aforementioned features extracted for a sliding window of size W and 50% overlap. Each data record is labeled with corresponding actual driving states. The RF algorithm fits randomly selected samples from the labeled training data to multiple decision trees using the principle of bootstrap-aggregating. It further predicts the class labels for unseen data by taking a majority vote among the trained decision trees and thereby offers better generalization performance over a single decision tree. The ParkGauge+ implementation incorporates a rather modest set of 10 trees with a maximum depth of 3 each.

Figure 3.10: Visualizing driving states by raw data.

Learning and Using the HMM Parameters

Given the observations made online, the HMM allows us to infer the (hidden) driving contexts, if the HMM model parameters discussed earlier in this section (under the paragraph ‘Modeling a Parking Process’) are fully specified. We briefly explain how we learn the HMM parameters (offline) based on the driving/sensing data collected in advance, as well as how to use them for online inference. For learning the HMM model, we apply the backward-forward recursion similar to Baum-Welch EM algorithm [71], but we opt for the Gibbs Sampling [36] technique for a more robust estimation: it makes better use of our dataset with driving states manually labeled and corresponding observations recorded.

Given the dataset obtained through our intensive tests, we apply common statistics to derive initial values for these HMM model parameters defined earlier. For example, the stationary probability of $\Pr(C_1 = \text{cruising})$ is obtained by looking at the frequency of the **cruising** context out of all observed driving contexts. Now given a sequence $\{o_1, o_2, \dots, o_T\}$ of observed driving states, the forward-backward procedure is used to compute the *forward posterior probability* and the *backward posterior probability*. The model is iteratively updated based on these probabilities and repeated until convergence to stable estimations of the model. To apply this model for inferring the driving contexts online, we make use of a modified Viterbi algorithm [72]; it identifies the most likely sequence of hidden driving contexts from a sequence of observations (driving states). At runtime, whenever a change of context is signaled, the timestamp is also recorded. In particular, the *Parked Time* is recorded if the context **cruising** changes to **parked**. The detection of **parked** has to be robust and reactive, as it is used to calculate **time-to-park**. Generally, the emission of **parked** is easily triggered by detection of the **walking** driving state. However, a continuous stay in driving state **pausing** may also lead to **parked**, handling the cases where the driver leaves the phone inside the parked vehicle.

3.3.2.4 Computing the Parking Characteristics

Given the context changing times recorded, we can compute the parking characteristics. While **time-to-park** is the difference between *Parked Time* and *Parking Arrival Time*, **time-in-cruising** and **time-in-queuing** simply accumulate the time spent in the corresponding contexts.

3.4 Empirical Evaluation

In this section, we describe the evaluation procedure, setup for data collection and parking experiments using our ParkGauge+ prototype as well as the evaluation results.

3.4.1 Data Collection and General Experiment Setup

For an effective evaluation, we have implemented ParkGauge+ as an Android-based application along with a back-end server. While the model learning procedure is done offline in the server using the collected dataset, all the online detections are performed by the ParkGauge+ client application running (at the background) in a smartphone, which periodically reports data to the server over a 3G/4G connection. We currently use a single Amazon EC2 instance on AWS cloud as our backend server.

Our experiments involved five different phones: Samsung Galaxy S4, HTC One M8, LG Google Nexus 5, Samsung Galaxy S3, and HTC One X. All these phones possess both accelerometer and gyroscope, while the first three of them also have a barometer. To validate the inter-floor height calculated by the barometer, we used a Bosch laser rangefinder DLE40 to obtain the ground truth. We collected data for three months in two different cities (primarily in Singapore and also in London). Six drivers of different nationalities and diverse driving habits contributed to this dataset and both petrol (Toyota WISH, VW Polo, Mazda 3) and electric (BMW i3) cars were used. One of these drivers included a co-author (of our publications that formed the basis of this chapter) who was aware of the methodology, while the others were volunteers; all of these drivers performed parking in the selected garages by following their naturalistic driving and parking preferences.

We have conducted the experiments in 4 different parking garages that were attached to popular shopping malls and specifically selected so that we get significant fluctuations in their occupancies during a day. These garages published their occupancies online¹, which we collect and use as ground truth. We performed two categories of designed experiments: i) parking among a few close-by garages in a sequence, and ii) parking in a single garage in a cyclical manner, i.e. enter, park, walk, unpark, exit, and

¹Data is obtained from LTA DataMall <https://www.mytransport.sg/content/mytransport/home/dataMall/dynamic-data.html>

repeat. On one hand, the former helps us to evaluate the robustness of ParkGauge+ in accommodating different parking situations: these garages were quite different in terms of number of floors (ranging from 3 to 12 floors), underground or aboveground, floor area and capacity per floor etc. On the other hand, the latter category enables us to evaluate the consistency of ParkGauge+ and also to verify the characteristics measured by ParkGauge+ against the ground truth during the same period of time. Each parking session took roughly 10 to 25 minutes, depending on the occupancy at that time. To capture the fluctuations in the demand for parking, the data collection was performed around the peak hours, i.e., 10:00-16:00 and 18:00-20:00.

3.4.2 Data Pre-Processing

We work with 2 kinds of data: (1) sensor data collected from smartphones during parking sessions, (2) parking characteristics determined by ParkGauge+ users. We note that evaluating the performance of occupancy *transfer learning* requires the latter to be organized into “local” and “external” datasets and this will be separately described in Section 3.4.5.

3.4.2.1 Pre-processing Raw Sensor Data

The sensor data obtained from smartphones is cleaned and smoothened to remove noise and outliers, as already described in Section 3.3.2.3. The cleaned data is used to extract features and train the driving state classifier and the HMM. 10-fold cross-validation is performed to ensure good generalization performance of these methods.

3.4.2.2 Pre-processing the Parking Characteristics Dataset

This dataset included both (a) a feature-rich set of parking characteristics computed by the ParkGauge+ clients and validated against corresponding ground truth, as well as (b) (availability, time-to-park) data pairs collected at few garages through manual observation. The former dataset is richer in features and very diverse since they are from different days and involving different drivers, albeit the number of observations per driver or vehicle and per garage and per day are small. However, since this is crowdsensed and inferred data, there are some chances of errors. The latter data is

collected for a specific garage for a long duration (e.g. 10:00-16:00) and recorded at least once in every 10 minutes, and this resulted in data of high quality and reliability, and nearly identically and independently distributed (i.i.d). This dataset is used to construct the regression model and gauge the parking occupancy. Therefore, the dataset is first cleaned to remove extreme values or outliers. We first calculate the Inter-Quartile Ranges (IQR) on all records and then filter and eliminate the values in 25% and 75% quartiles that exceed the IQR. Subsequently, we normalize all the numeric features in all the data records using min-max normalization before running the regression procedure.

Feature Selection

After pre-processing, we apply a feature selection algorithm to identify a significant feature subset i.e., the best features (parking characteristics) useful to infer the parking occupancy. Although this could be done by a filter or wrapper method, we employ Correlation Feature Selection (CFS) [73]. This method evaluates the inference quality of a subset of attributes by evaluating the individual predictive ability of each feature in addition to the degree of redundancy between these features. Running CFS on our training dataset with 10-fold cross validation helped to arrive at the 7 parking characteristics listed in Table 3.1 as the selected features. In our original data, additional fine-grained parking characteristics were available such as the number of floors traversed, left/right turns count. However, these characteristics were excluded as they did not possess significant predictive ability within this dataset. With larger datasets arriving from a large-scale crowdsensing deployment, such feature selection is highly useful.

Splitting The Dataset for Training and Testing

After pre-processing, we evaluate several regression methods. For each method, we train the model using our training dataset and perform 10-fold cross validation to understand their generalization performance. Finally, we test the trained model with a testing dataset and report the performance.

For the experiments outlined in this chapter, we evaluated the methods with data collected for several days from the same driver as well as from different drivers. Finally, we intentionally mixed the data ParkGauge+ reports from actual parking sessions with the manually observed dataset, since this approximates the data inputs to a real-world

Figure 3.11: (a) Evaluation of PAD performance as start/give-up-parking events estimated vs. actual at two parking garages CP1 and CP2 (b) CDF for error in time-to-park estimates.

crowdsensing system. This dataset consists of 103 records, of which 37 records are manual observations with a high fidelity and the rest are crowdsensed by ParkGauge+ during parking experiments. The manual observations had empty features which were marked as missing values, but valid features were still considered. This dataset was further divided into training data and testing data. We performed both 10-fold and leave-one-out cross validation approaches to ensure generalization performance. The testing data was obtained by randomly sampling 30 observations without replacement, resulting in a 30-record testing dataset and a 73-record training dataset.

3.4.3 Evaluating ParkGauge+ Sensing Core

Here, we evaluate how ParkGauge+ performs in the inference of driving states, contexts and parking characteristics.

3.4.3.1 Parking Arrival Detection Performance

As described in Section 3.3.2.1, vehicles stuck in traffic on the roads close to a garage entrance may falsely trigger **start-parking**. However, as shown in Fig. 3.11a, this happens only in few cases, viz., only 4 instances vs. 58 actual parking sessions at a garage CP1, and only once for another garage CP2 against 22 actual parking sessions made there. The PAD is able to signal **give-up-parking** in all of these cases. This shows that our approach is generally robust and such exceptional cases do not affect the overall performance of ParkGauge+.

Figure 3.12: Raw data traces corresponding to three parking trials with short, medium and long values of time-to-park. We provide full labels only to the long trace for better readability, but the feature-context correspondence can be easily discerned for the other two traces also.

Table 3.3: Precision and recall results for the performance of driving state detector.

Driving state	Precision	Recall
proceeding	92.86	89.66
accelerating	94.78	90.08
decelerating	90.85	89.58
braking	95.45	92.65
pausing	98.33	98.88
walking	96.85	95.35
turning	96.23	96.23
Overall	95.05	93.20

3.4.3.2 Inferring Driving States and Contexts from Raw Data

To provide a rough idea about how ParkGauge+ operates in field, we use three sets of raw sensor readings plotted in Fig. 3.12 to demonstrate the inferred driving states and contexts. These three parking trials differ in their values of **time-to-park** reported by ParkGauge+: (1) The top-left one took only 140 seconds as the vehicle arrived at a parking garage with a rather low occupancy (in the morning), and most of the time was spent mainly on cruising and going two-levels downwards (indicated clearly by barometer readings). (2) The top-right one took about 2 more minutes. According to ParkGauge+’s detection, the vehicle already faced a small queue before entering the parking garage, and as the occupancy was relatively high, it had to get one more level down in order to find a proper parking lot. According to our ground truth observation, this was the period right before noon, when the parking occupancy was steadily increasing. (3) The bottom case was a long parking trial. The vehicle arrived at the parking garage with almost full occupancy, so it was first hit by a long queue at the entrance, then it spent quite a long cruising time in finding a parking lot even after getting three levels down. This trial took place during the peak hour when people drive to the shopping mall to get their lunch.

Furthermore, we evaluate the performance of the Driving State Detector (DSD) in more detail. In fact, the Random Forest based classifier adopted by the DSD detects the various driving states with an overall precision of 95.05% and a recall of

Figure 3.13: Precision and recall for driving state detector.

93.03% as shown in the Table 3.3. From this table and Fig. 3.13, we note that **pausing** is detected very accurately with a precision of 98.33% and a recall of 98.88%, while **walking** and **turning** are recognized with a slightly lower precision and recall at around 96%. The detection of **braking** and **accelerating** performs well with (precision, recall) of (95.45%, 92.65%) and (94.78%, 90.08%) respectively. But, the recall for **proceeding** and **decelerating** are slightly inferior at around 89%. This could hint that the features extracted from noisy smartphone sensor data may not have sufficient predictive power to *perfectly* discriminate these two specific driving states that depend on the vehicle dynamics. However, we also note that these two driving states seem to have a much lower influence on the Hidden Markov Model of DCD, in comparison to states such as **pausing**, **braking** or **walking**. Therefore, any occasional mis-detection of **proceeding** or **decelerating** is quite unlikely to affect the driving context detection or the overall performance of ParkGauge+ in inferring parking occupancy.

3.4.3.3 Accuracy of Driving Context Inference

We omit the performance evaluation for driving state inference and directly evaluate the accuracy of estimating driving contexts, because driving contexts are directly related to ParkGauge+’s final performance in deriving parking characteristics and thus inferring the occupancy. To evaluate the accuracy of estimating a driving context, we first identify the time intervals spent in that context according to our ground truth observation. Then, we compare the corresponding output of ParkGauge+ to calculate

the *percentage* of intervals when a correct estimation is made. Assuming an ergodic stochastic process behind the estimation, this percentage exactly indicates the success rate in detecting this driving context.

The other goal of this evaluation is to choose the best window size W used to estimate driving state and context. In fact, we initialize W to 5s, as this is the smallest value to accommodate a driving state (e.g., **turning**), and then try with other multiples of 5s. As shown in Fig. 3.14, while it is expected that the latency grows almost linearly with W , the highest accuracy is achieved at $W = 10$ s. The accuracy first increases with W as ParkGauge+ gets more information to make a decision, but it starts to decrease later as the “cost” of making a wrong estimation grows with W . As a result, we fix $W = 10$ s in our final implementation.

3.4.3.4 Accuracy of time-to-park Estimation

Estimating **time-to-park** is an important functionality of ParkGauge+. As shown in Fig. 3.11b, the estimation errors range from 5s to around 30s. While extreme values are rare, most errors are around 18s. Since the normal values of **time-to-park** are in the order of minutes (or tens of minutes in the worst case), we deem ParkGauge+’s performance in estimating **time-to-park** very accurate. The few cases of extreme error values were primarily due to the additional latency incurred by DCD in detecting **parked** and a slightly early detection (before actual arrival) of **start-parking** by the PAD under erroneous GPS localization within urban canyons. In fact, as the error mean is related to the window size W (it is systematically biased due to the detection latency), we could further reduce the estimation error by subtracting W from each estimated **time-to-park**. This would bring the error down to the range of $(-5, 20)$ s.

3.4.4 Evaluating Parking Occupancy Inference

Here we evaluate the performance of different regression methods to estimate occupancy from parking characteristics, with only the “local data” as described in Section 3.4.2.2.

Figure 3.14: Detection accuracy and latency for driving contexts driving, cruising, parked, queuing and floor-change under various window sizes W with a step size of 5s, followed by start-parking and give-up-parking under different geofence radii with a step size of 5m

3.4.4.1 Regression Performance for Occupancy Inference

As explained in Section 3.3.1.1, we use regression to recover the functional relationship between occupancy and temporal parking characteristic **time-to-park** extracted from the data. We evaluate using the training and test datasets described in Section 3.4.2.2. We compare four regression approaches, namely: (1) Simple Linear Regression (SLR)

with only **time-to-park** feature, (2) Multiple Linear Regression (MLR) with all 7 features (parking characteristics) listed in Table 3.1, (3) Artificial Neural Network (ANN) using 10 hidden layers, and (4) SVR with 5 different kernel functions, namely Linear, Polynomial (quadratic, cubic), RBF and PUK.

To compare the different regression methods, we use the Normalized Root-Mean Squared Error,

$$NRMSE = \sqrt{\frac{\sum_{i=1}^n (p_i - a_i)^2}{n}}, \quad (3.3)$$

the square root of the mean of squares of errors between the normalized actual values $\{a_i\}$ (as discussed in Section 3.4.2.2) and the corresponding inferred values $\{p_i\}$.

Figure 3.15: Comparison of different regression methods

From Fig. 3.15, one observes that linear regression approaches (MLR and SLR) performed reasonably well. The NRMSE is high, but it is still comparable with SVR. ANN does not perform very well with this dataset. Using the hidden layers, it seems to over-fit the model to the training data and hence the test data results in higher errors. However, SVR provides good performance with different kernel functions. Linear kernel offers slightly worser error performance, whereas Polynomial (both Quadratic and Cubic) and RBF kernel functions offer superior performance. But, it is clear that SVR with PUK kernel offers the lowest NRMSE (0.0993) and hence, the best generalization performance.

A key observation from this evaluation is that SLR performs decently (NRMSE 0.2106) even when using **time-to-park** as the only input feature. However, the accuracy obtained is more than doubled (NRMSE 0.0993) when we perform SVR with PUK

Figure 3.16: Comparison of ParkGauge+ performance with a simpler approach measuring time-to-park from GPS loss.

kernel on a higher-dimensional training data with additional features (parking characteristics) such as time-in-queuing as listed under Section 3.3.1.1. This justifies the multi-layered hierarchical methodology adopted by ParkGauge+.

To clearly distinguish the performance of ParkGauge+, we compare how these regression methods perform on a simpler sensing scheme measuring the time-to-park by detecting loss of GPS signal upon entering the garage and detecting the transition to a pedestrian state (i.e. walking or not) upon parking. From Fig. 3.16, we can see that ParkGauge+ (NRMSE=0.0993) significantly outperforms this alternative approach (NRMSE=0.3221), especially with our chosen method of using SVR with PUK kernel (by a factor of more than 3 times).

3.4.4.2 Tuning The Parameters of SVR(PUK)

Since we have convincing results that help us choose SVR(PUK) as the preferred approach to gauge parking occupancy, we now tune the parameters of the PUK kernel function, namely ω and σ , that control its shape and thereby affects the inference output of SVR(PUK). We vary each parameter across a range $[0, 100]$ while keeping the other parameter fixed at a value 1. In each iteration, we train a SVR(PUK) model and check its generalization performance using 10-fold cross validation. In Figure 3.17 (left plot), one observes that the lowest NRMSE is obtained at $\omega = 2$ when σ is fixed at 1. In the right plot, one observes that the best performance (0.0993) is obtained at $\sigma = 1$

Figure 3.17: Determining the values of Pearson parameters ω and σ that yield the best performance for SVR(PUK). Parameter values on x-axis are uniformly spaced and clipped at 10 to enable a clearer presentation.

when ω is fixed at 1. Thus, we choose the common point ($\sigma = 1$, $\omega = 2$) as the Pearson parameters for SVR(PUK) in inferring parking occupancy for ParkGauge+.

3.4.5 Evaluating Parking Occupancy Transfer Learning

Here, we evaluate how ParkGauge+ performs in inferring parking occupancy by applying transfer learning.

3.4.5.1 Preparing the Transfer Dataset

We consider a known source dataset from the same garages that we evaluated previously in Section 3.4.2.2. Here, we can include both the original training and testing data to form the D_S which contains 103 records. Now, we consider a target parking garage within the same urban district, and having a similar structure (attached to a shopping mall and with 2 underground floors). We collected data for 22 parking sessions across 2 working days spanning the peak hours. After pre-processing as in Section 3.4.2.2, we split this data into a 8-record target training dataset and a 14-record target testing dataset. This ensures that the D_S is significantly larger than D_T by one order of magnitude, which is representative of real life situations. These datasets contain all the characteristics listed in Table 3.1 except for an additional ‘floor change count’ that represents the number of floors changed while the vehicle is cruising indoor. This is

because the CFS feature selection algorithm [73] recognizes ‘floor change count’ as a strong feature, due to its correlation with **time-to-park** and the parking occupancy. A normalized ‘floor change count’ also helps to iron out the influence of structural differences between source and target garages.

3.4.5.2 Performance of Traditional Regression Learning Approaches on the Target Dataset

We first evaluate the regression performance on the small D_T (“local” data) with traditional machine learning approaches such as Multiple Linear Regression (MLR), and SVR with different kernel functions: Linear, Polynomial(degree 2,3,4) and PUK. As illustrated in Figure 3.18, the results indicate that some of the traditional approaches that try to discover linear relationships in the dataset, such as SVR-Lin (NRMSE 0.167) and MLR-tar (NRMSE 0.219) perform relatively well. SVR-PUK (NRMSE 0.221) and SVR-RBF (NRMSE 0.189) also perform well and achieve reasonably low NRMSE with the testing dataset. However, we can see that SVR with higher degree polynomial kernels such as SVR-Poly(3) (NRMSE 0.378) and SVR-Poly(4) (NRMSE 0.711) perform worser due to overfitting.

Figure 3.18: (1) green bars on the left indicate how the traditional learning methods perform on a small target training dataset D_T from a new garage, (2) blue bars on the right show transfer learning methods on a combined dataset D

Figure 3.19: Evaluation of *two-stage TrAdaBoost.R2* to find (a) the optimal value of parameter S i.e. the number of stage-2 iterations, and (b) the optimal base regression learner method, that could be adopted for deployment.

3.4.5.3 Performance of Transfer Learning Approaches on the Combined Transfer Dataset

We now evaluate the learning approaches using the combined transfer dataset D . We use D to evaluate the performance of MLR and *AdaBoost.R2* (labeled as aBst.r2), as well as two *instance-based* inductive transfer learning approaches, namely *TrAdaBoost.R2* (tr-aBst.r2) and *two-stage TrAdaBoost.R2* (2stg-tr-aBst.r2). We find that *AdaBoost.r2* (NRMSE 0.287) and *TrAdaBoost.R2* (NRMSE 0.247) do not offer attractive results. However, the two-stage transfer learning approach using MLR as $bLrn$ and operating on the transfer dataset D performs best (NRMSE 0.150). It is also noteworthy that MLR operating on D as the main learner (MLR-trf) (NRMSE 0.159), significantly *improves* over the traditional MLR-tr (NRMSE 0.219). This demonstrates that a key aspect of the transfer learning methodology, namely, the approach of strengthening a small D_T by combining it with a rich D_S dataset can improve the parking occupancy inference performance, especially when they have a similar distribution. Selectively boosting or damping the weights of instances depending on their similarity with the target distribution does result in improved performance as we can discern from the promising performance of *two-stage TrAdaBoost.R2* utilizing cross-validated boosted models.

Figure 3.19(a) illustrates the effect of S on the performance. We obtain the lowest testing NRMSE 0.137 at $S=20$, however the training error is slightly higher than the

minimum. However, at $S=15$, we obtain the lowest training NRMSE and the second lowest testing NRMSE (0.150). In general, we can discern that the best occupancy inference performance for this target parking garage is obtained between $S=15$ and $S=20$. Any further increase in S leads to severe over-fitting and higher testing errors. At lower values of S , we would have a faster training, but inferior performance. Since larger S would lead to both increased computation time and weaker inference quality, we can generally fix the value $S=15$ for the given garage. We also note that for each new target garage, we may have to repeat the transfer learning procedure and this analysis to find out the optimal value of S which could be different from $S=15$. However, transfer learning is still beneficial as it minimizes the expensive data collection efforts for training.

3.4.5.4 Choosing the Base Regression Learner for Boosting

The performance of a boosting algorithm often depends on the right choice of a base learner $bLrn$. To ensure optimal transfer learning performance, we evaluate the below regression methods with good temporal efficiency as $bLrn$ for *two-stage AdaBoost.R2* by training on the combined dataset T : Multiple Linear Regression (MLR), Linear Regression Tree (LinRegTree), Extreme Learning Machine (ELM) [74] and Support Vector Regression with Linear (SVR-Lin) and PUK (SVR-PUK) kernels. As shown in Fig. 3.19(b), we obtain the best NRMSE.Test and fastest training time (0.150, 0.5s) with our testing dataset when we use MLR as the base learner. LinRegTree (0.176, 20s) and ELM (0.182, 1.5s) offer slightly inferior performance, while SVR-Lin (.350, 1117s) and SVR-PUK (.344, 512s) do not perform well here. Since MLR offers the best NRMSE and excellent temporal efficiency, we choose MLR as the preferred base regression learner.

3.5 Discussion

In this section, we discuss some key considerations, special cases and challenges to be deliberated with regard to practical deployments of ParkGauge+.

3.5.1 Minimal Data Requirements

Based on above experiments in Section 3.4, we can briefly summarize the data sizes required to make useful inferences. First of all, the supervised learning-based occupancy inference of ParkGauge+ offers reasonably good performance with ≥ 50 sensor data traces from parking sessions at a particular garage. For transfer learning, as mentioned in Section 23 and 3.4.5.1, we expect a small D_T of at least 5-10 traces for the target garage (preferably spread across different times of day and days of week), and a larger D_S of size at least an order of magnitude higher (≥ 50 traces) from other similar garages. However, even such minimal data expectations could potentially be relaxed by applying *zero*, *one* or *few*-shot learning methodologies [75], [76], [77]. Although such learning methods mainly find success in computer vision problems, we believe that they could be adapted to solve the parking occupancy inference regression problem too.

3.5.2 Impact of garage restrictions

Some of the garages where ParkGauge+ could be adopted may offer some additional restrictions that may challenge the occupancy inference performance. These include and are not limited to the following:

- *temporal restrictions*: Some garages may impose restrictions on available parking spaces based on time-of-day and/or day-of-week. For instance, some spaces or zones or even entire floors may be reserved only for season parking pass holders during particular time periods. Although ParkGauge+ does consider **time-of-day** and **day-of-week** as relevant parking characteristics, in order to properly handle such cases, the Occupancy Estimator will need to be trained with exact floor level information. Optionally, the indoor localization methodology proposed in Chapter 4 may also be integrated to selectively distinguish between parking sessions in such restricted areas and freely accessible areas and calibrate the occupancy inference methodology. The smartphone app may also introduce direct user inputs to specify the season parking privileges of the vehicle.
- *vehicular diversity*: Some garages may impose restrictions based on vehicle type, such as electric vs. non-electric, compact car vs. SUV and similar. In such garages, ParkGauge+ must be prepared to handle the vehicular diversity and

the resulting impact on parking occupancy. In such cases, ParkGauge+ cannot treat the entire parking garage or even a floor as a uniform entity, and must rather distinguish them based on location, floor and vehicle type. If the vehicle type is made known (either specified through user input or learnt otherwise), ParkGauge+ can learn the relationship between vehicle type and occupancy and the associated **time-to-park**, and use it as an additional parking characteristic in the occupancy inference process.

3.5.3 Impact of special events

Whenever there are any special events, the traffic and parking demand at/around a garage can undergo a major change from their normal levels. This is especially the case for mega events. However, this can cause the occupancy inference performance of ParkGauge+ to deteriorate given our assumption of an ergodic process (which is usually valid, but not in such special cases). Nevertheless, we could also expect a larger than usual participation rate of ParkGauge+ users as human drivers generally tend to use or depend on assistive services more often when their mobility becomes difficult. This can result in the acquisition of larger user data samples to adaptively train ParkGauge+ and correctly infer the occupancy based on the latest parking demand patterns. Therefore, although performance deterioration can occur in the short term, the system could cope with it soon after and recover to deliver the full quality of service.

3.6 Summary

In this chapter, we have proposed a mobile crowdsensing system called ParkGauge+ that delivers real-time occupancy information about indoor parking garages to its users. The key motivation behind ParkGauge+ is our observation that users of many indoor garages may lack real-time occupancy information until they arrive, which can lead to long queues and traffic congestion. On the other hand, the methods adopted by existing proposals for outdoor parking mostly involve direct counting of occupied parking lots, which is not practical for large indoor garages in an infrastructure-free manner due to the lack of sufficient indoor localization capability without relying on GPS or WiFi or similar infrastructure. To this end, our design of ParkGauge+ innovates in

using inertial sensing data to infer parking occupancy with reasonably low errors by computing parking characteristics such as **time-to-park** and number of **braking** or **turning** instances, and applying several machine learning methods without any need for precise indoor localization. Through extensive experiments with a reference implementation, we have demonstrated the very promising performance of ParkGauge+. To ensure the scalability, we apply boosting-based transfer learning methods by reusing existing datasets and gauge the occupancy of new garages without significant additional data acquisition. Our evaluation produced promising results that demonstrate the merit in this approach, although reusability of each source dataset does depend on the domain and distribution. Since our current transfer learning method still requires some small training data, we plan to explore zero-shot, one-shot and few-shot learning methods using zero, one or only few training examples. Finally, we strongly believe that the idea behind our design of ParkGauge+ can be extended to outdoor parking and also other types of occupancy detection scenarios, hence we are on the way to expand its application scope.

Chapter 4

ParkLoc: Light-weight Graph-based Vehicular Localization in Parking Garages

In Chapter 3, we presented ParkGauge+ to tackle the problem of estimating the occupancy of an indoor parking garage even in the absence of infrastructure support such as GPS and/or WiFi, and without any need for precise indoor localization. As a matter of fact, it is the lack of such supportive infrastructure that makes the indoor vehicular localization a rather difficult problem to tackle in general, especially when the vehicle is underground (such as in the basement of a parking garage). Unfortunately, this limits the deployment of indoor-location-based applications or services for vehicles.

As we have seen in Chapter 2, the state-of-the-art GPS-free indoor localization approaches generally rely on some *other* infrastructure support such as WiFi access points or BLE beacons, whereas the smartphone-only proposals mostly require some significant data collection and training efforts for each garage. In this context, we propose ParkLoc, a novel light-weight graph-based smartphone-only solution for vehicular localization in indoor parking garages, which is also able to generate a map graph of the garage by employing a semi-supervised variant of the popular GraphSLAM algorithm from robotics research. The detailed experimental results presented in this chapter demonstrates the effectiveness and performance of our proposed methods.

4.1 Introduction

Most (human) drivers prefer to use navigational aids while driving on unfamiliar routes and during uncertain situations such as searching for available parking space at a large indoor parking garage. Many drivers also find it difficult to recall/recognize their parked locations while returning, making them wonder: *Where did I park?* A 2014 survey of 2000 drivers in USA found that 52 percent of participants reported “forgot where they parked” as the most common driving embarrassment [78] ; this is often due to the absence of visually distinguishable landmarks or furniture in large garages where most areas look alike. Since GPS signals are mostly unavailable indoors, garage operators may try to help guide drivers by installing dynamic directional signs indicating their zone-wise parking availability or by tagging each zone with QR codes, RFID or NFC. Nevertheless, the necessary infrastructure can be cost-prohibitive (in particular for RFID or NFC [79, 80]), especially to those in developing countries earning lower parking revenues. Such systems may even require manual user intervention such as stopping the vehicle to scan or tap, and they can incur high energy consumption in case of automatic QR code detection through smartphone camera. To avoid any dependency on expensive infrastructure and to ensure a seamless navigational experience, it will be ideal if such indoor location-aware guidance capability can be built into the driver’s smartphone.

To address the general indoor localization problem, researchers have studied many GPS-free alternatives, including wireless fingerprinting approaches, inertial dead reckoning, geo-magnetic fingerprinting or visual odometry. However, these approaches may not perform well in highly constrained and non-instrumented environments such as indoor parking facilities. Whereas WiFi-based methods [81] are effective in places such as shopping malls or food courts, most indoor parking garages have their WiFi signals either unavailable or highly attenuated. Dead-reckoning (DR) methods [82] are promising for pedestrians, but they rely on step-counting while walking and thus inapplicable to driving or parking (when a driver remains static within the vehicle). Geo-magnetic fingerprinting-based methods may not perform well either in a moving vehicle because the in-vehicle noise interferes with the geo-magnetic signature of a location [83, 84].

In the meantime, autonomously driving vehicles have adopted robotic localization algorithms with high-quality vehicle-mounted sensors [54] to achieve an amazing

centimetre-level accuracy. Unfortunately, smartphones lack such exteroceptive sensing capabilities and are hence unable to offer comparable results using only noisy MEMS sensors. Recent researches on vehicular localization strengthen the traditional outdoor-localization techniques with either inertial dead reckoning, multi-source sensor fusion or machine learning-based landmark detection. CarLoc [59] demonstrates how outdoor localization accuracy can be improved by combining visual roadway landmark information and odometry/orientation estimates. VeLoc [15] and VeTrack [16] propose a smartphone-only particle filter-based method with indoor garage map constraints and turn/bump detection based on supervised machine learning. Whereas the former relies on GPS and is designed for outdoor situations, the latter requires a detailed map and a substantial amount of training data for each garage.

Since all the current proposals expect large training data or infrastructural support, it is imperative to develop a light-weight smartphone-only localization system that can produce meaningful results with no or minimal training/tuning effort per parking garage. To this end, we propose *ParkLoc*, a new parking localization system that performs online localization on a simple yet effective topological planar graph representation of the map, either given beforehand or learnt from trajectories. At the heart of ParkLoc’s localization approach lies an approximate graph-matching method. The intuition is that the smartphone gyroscope data obtained at turn maneuvers help us naturally form a sparse planar graph (we term this *trajectory graph*), which can be matched onto a supergraph (we term this the *map graph*) that represents all the navigable paths within the garage. Essentially, we formulate the indoor vehicle localization problem as a *subgraph isomorphism problem* [62, 85]. In general, this problem is known to be NP-complete [62]; fortunately, our application scenarios restrict us to parking map/trajectory graphs that are naturally sparse, directed and of low cardinality, allowing the computations to be done quickly in practice. Furthermore, the map graph can be learnt via a semi-supervised version of GraphSLAM [53], albeit without the rich vision or ranging sensor suite typically used to detect landmarks in robotics.

As we can discern from Fig. 4.1, drivers following similar indoor trajectories tend to make a sequence of turn maneuvers with a very similar temporal distribution. This motivates us to exploit such spatio-temporal signatures to enhance graph matching, localization and mapping, by taking these inherent turning points (readily detectable

Figure 4.1: (left) Raw gyroscope data from 3 trips (different vehicles) at an indoor garage, following the same trajectory. Positive and negative peaks indicate left and right turns, respectively. (right) Histogram of the temporal distribution of turns in a trajectory graph, after pausing-adjustment. Labels a,b,c,d represent pose nodes where left/right turn maneuvers are made.

via inertial sensing) as *intrinsic landmarks*. So our major contributions are:

- A novel idea for localizing a vehicle within a parking garage by taking advantage of the spatio-temporal characteristics of turn maneuvers.
- A method to exploit the inherent planar graph structure of a trajectory obtained from a sensor trace and use approximate graph matching methods for online localization.
- A semi-supervised learning approach to learn a map of the garage given a set of smartphone inertial sensor traces and a few seeds.
- A system that demonstrates the potential of real-time graph-based localization

Figure 4.2: System architecture of ParkLoc.

and map learning using only commodity smartphones via a full scale evaluation.

The rest of this chapter is organized as follows. In Section 4.2, we present an overview of the ParkLoc system architecture and a detailed technical explanation of the methodology for sensor data processing, trajectory graph generation, graph matching and mapping. Further, we evaluate the proposed ideas and methods in Section 4.3 and summarize the chapter in Section 4.5.

4.2 Methodology

In this section, we first provide an overview of the main architectural components of ParkLoc and subsequently explain each component and its associated methods.

4.2.1 ParkLoc Architecture

As shown in Fig. 4.2, ParkLoc consists of three major architectural components: (a) the Trajectory Grapher that processes the smartphone sensor data stream and generates a trajectory graph in an online fashion (b) an offline GraphSLAM-based Mapper

that processes the trajectory graphs from multiple trajectories in a semi-supervised manner considering a set of seed landmark nodes and generates (updates) a map graph representing the navigational structure of the parking garage, and (c) an Online Graph Matcher & Localizer component that finds a match between the current trajectory graph and the map graph to localize the vehicle’s current position to the nearest path segment. The estimated *vehicle location* includes the 2-d position (in terms of WGS84-based latitude and longitude) and heading (azimuth w.r.t. geographic north) of the vehicle, and is associated to an arbitrary point lying on edge of the map graph. We can understand the general process using the example graphs illustrated inline within Fig. 4.2. The trajectory graphing and matching are performed online within a smart-phone application, while the map graph is generated offline in a backend server. When a vehicle arrives and approaches a garage for parking, ParkLoc loads the corresponding map graph (if available) from a graph database to be used in online localization. The generated trajectory graphs are stored into a database to perform offline mapping.

Remarks on Integration with Pedestrian Navigation Systems

Drivers using ParkLoc would be most benefited if they could obtain a seamless end-to-end localization experience as they transit to a pedestrian mode after parking and return to find the parked vehicle. However, we restrict the scope of this chapter to the vehicular mode due to the page limit and the proliferation of pedestrian indoor localization proposals available. Nevertheless, for the sake of completeness, here we briefly explain the application cases where ParkLoc may be integrated with other schemes, while omitting a full-scale evaluation of the same. We consider three well-known pedestrian indoor localization schemes viz. PlaceLab [86] and Zee [29] both using WiFi, and UnLoc ([43], [44]) that can work without WiFi. For simplicity, we assume that outdoor localization (both in-vehicle and as pedestrian) is handled primarily through Assisted GPS (or A-GPS) and where unavailable, through augmented mobile systems such as CarLoc [59] or WoLoc [30]. Assuming that we can have an independent Mobile Localization Manager (MLM) system that is authorized to switch between the most suitable localization scheme depending on the currently active and/or next anticipated mobility context, the two major user-level application scenarios can be handled as follows.

- *Vehicle to Pedestrian Transition*: Assume that the driver is driving a vehicle on-road (assisted by A-GPS or CarLoc) and wants to park his/her vehicle. Upon

detection of arrival at a recommended parking garage [87], MLM switches to ParkLoc to provide the localization support. ParkLoc locates the vehicle until it is parked and stores the parked vehicle location. As the driver leaves the vehicle, the MLM detects a pedestrian context and switches to an indoor scheme which is viable without WiFi (e.g., UnLoc) and hands over the vehicle’s parked location as the start position. As she proceeds to a building with infrastructure support for localization, MLM may continue with UnLoc or switch to another scheme (e.g., PlaceLab or Zee) if required.

- *Pedestrian to Vehicle Transition*: As the driver returns from the indoor building to the vehicle, we expect the roles to reverse. That is, MLM can switch from the currently active in-building localization scheme to UnLoc until the user is guided to the location of the parked vehicle, while anticipating a switch to ParkLoc upon arrival at the vehicle. Once an in-vehicle context is detected, ParkLoc is activated until the vehicle exits the garage. Finally, a valid outdoor scheme can be started to maintain localization on the road.

4.2.2 Sensing: Preparation and Processing

Here we outline the key steps required to prepare and process the sensor data to achieve the objectives of ParkLoc.

4.2.2.1 Parking Arrival Detection

Since ParkLoc is specifically designed to work for indoor parking garages, it needs to be started only when arriving at the garage. To achieve this in an autonomous mode, we adopt a coarse GPS-based geo-fencing technique [87] to trigger and start ParkLoc upon the vehicle’s arrival at the outdoor ingress streets to the garage.

4.2.2.2 Smartphone-Vehicle Axes Alignment

We first smoothen and mitigate the effects of noise on the raw sensor data by using an exponential smoothing (low pass filter) step such that for every i th raw sensor reading r_i , the corresponding smoothed value $\hat{r}_i = \beta r_i + (1 - \beta)\hat{r}_{i-1}$ where β is a smoothing constant (we use $\beta = 0.1$). Furthermore, we align the smartphone axes (which may

be placed at any arbitrary 3D attitude) to the vehicle-aligned axes, by computing a rotation matrix and projecting the measured values onto the corresponding longitudinal (x), lateral (y) and vertical (z) axes of the vehicle[88], similar to [89]. We may also consider the alternative approaches [26, 33, 47]. When the vehicle makes a turn, the variations in angular velocity ω (*yaw rate*) are concentrated around the vehicle z -axis. Hence, in ParkLoc, we can safely restrict the use of ω values to the z -axis.

We also note that there could be a slight disturbance to this process if the smart-phone is moving (especially due to user action). However, this effect is only momentary as the orientation will be immediately re-computed and axes aligned at the next window. There could be some positional error within a window if there's significant phone movement within that duration, but in practice, the system recovers very quickly.

4.2.2.3 Pausing Adjustment

In order to detect whether vehicle is actually moving (driving) or not (pausing), we utilize a C4.5 decision-tree based classifier [90] trained on time-domain features such as mean and variance of accelerometer readings obtained in sliding windows of 1 second duration each with 50% overlap. We continuously identify such short sections where the vehicle is pausing i.e. the temporary stops during a trip. Whenever we identify such sections, we remove the sensor traces for the entire time-frame the vehicle is idle. This helps us get accurate estimates of the driving time between nodes, which is crucial to achieve good graph matching performance as explained in Section 4.2.4. Alternatively, we may also use Dynamic Time Warping (DTW) with Dynamic Programming to align [47] or normalize [91] the sensor data traces, however we find that our proposed decision-tree based pausing adjustment method is simpler, more efficient and accurate enough for our application.

4.2.3 Online Trajectory Grapher

In this section, we describe the procedure to generate a trajectory graph from raw sensor observations. We first describe how to estimate the steering angle and velocity of the moving vehicle. Then, we describe the motion model to obtain a rough position estimate of the vehicle using dead-reckoning. Finally, we describe how to generate a trajectory graph from the sequence of position estimates.

4.2.3.1 Vehicular Motion Model

In order to calculate the vehicle position estimates, we use a motion model based on dead reckoning over a stream of inertial sensor data which is smoothed, windowed, phone attitude-invariant and pausing-adjusted. In order to estimate the vehicle orientation, we utilize the yaw-rate values obtained by the gyroscope reading projected onto the vehicle-aligned z -axis of the smartphone. Since gyroscopes typically accumulate a bias that deteriorates their usability, we compensate this by first identifying the current bias value by computing the mean of gyro readings during time intervals when we observe very low variance and subtracting this gyro bias from all future readings. We repeat this process periodically, and we integrate the values over a sliding window of size $W_g = 3$ seconds with no overlap. Let $\Delta\theta$ be the orientation change from previous pose.

We update the velocity $\mathbf{v}$ for each window as: $\mathbf{v} = k[\tilde{v}_x \ \tilde{v}_y]$ where k is a calibration parameter dependent on the vehicle and mobile device, $\tilde{v}_x$ and $\tilde{v}_y$ are the vehicle-aligned (refer Section 4.2.2.2) velocities computed by integrating the accelerations obtained during Δt , the pausing-adjusted time elapsed since the last step (ideally, $\Delta t \approx W_g$) and speed $v = \|\mathbf{v}\|_2$. Let the current vehicle pose after i th step (or window) be represented by $\mathbf{x}_i = [x_i \ y_i \ \theta_i]$ and the preceding pose by $\mathbf{x}_{i-1} = [x_{i-1} \ y_{i-1} \ \theta_{i-1}]$. We use a relative metric coordinate system with the initial pose as the origin. Accordingly, we initialize the first pose as $\mathbf{x}_0 = [x_0 \ y_0 \ \theta_0] = [0 \ 0 \ 0]$. Any future pose thus calculated, can be transformed into a desired world coordinate system (e.g. WGS84) based on the prior knowledge of the true initial pose in such a system. To compute the new pose, we add the incremental pose transformation $\Delta\mathbf{x}$ to $\mathbf{x}_{i-1}$ considering Gaussian noise $\mathcal{E}$. We repeat this procedure as long as the vehicle is moving.

$$\begin{aligned} \mathbf{x}_i &= \mathbf{x}_{i-1} + \Delta\mathbf{x} \quad \forall i \\ \Delta\mathbf{x} &= \begin{bmatrix} v\Delta t \cos(\theta_i + \Delta\theta) \\ v\Delta t \sin(\theta_i + \Delta\theta) \\ \Delta\theta \end{bmatrix} + \mathcal{E} \end{aligned} \quad (4.1)$$

Whenever required, we transform the pose estimate $\mathbf{x}_i$ to a standard WGS84 $\langle \text{latitude}, \text{longitude}, \text{bearing} \rangle$ location representation $\langle \text{Lat}_i, \text{Lng}_i, \text{Brng}_i \rangle$ as follows. Let d be the Euclidean distance between pose $\mathbf{x}_i$ and any previous pose $\mathbf{x}_j$ with known

world coordinates, and R_e the radius of Earth.

$$d = \|[x_i \ y_i] - [x_j \ y_j]\|_2 \quad (4.2)$$

$$\begin{aligned} Lat_i &= \arcsin(\sin(Lat_j) \cos(d/R_e) \\ &\quad + \cos(Lat_j) \sin(d/R_e) \cos(\theta_j)) \\ \delta l_a &= \sin(\theta_j) \sin(d/R_e) \cos(Lat_j) \\ \delta l_b &= \cos(d/R_e) - \sin(Lat_j) \sin(Lat_i) \\ Lng_i &= Lng_j + \arctan(\delta l_a, \delta l_b) \\ Brng_i &= \theta_i \end{aligned} \quad (4.3)$$

4.2.3.2 Generating a Trajectory Graph

Given the vehicle poses estimated by the above motion model, we generate a trajectory graph. We note that in ideal cases, this graph should be consistent with the structural constraints of the paths a vehicle can navigate inside the parking garage i.e., very similar in shape to a subset of the map or floor plan. However, in reality, the trajectory can become deformed due to external causes such as the inherent sensor measurement and process noises in the system.

Formally, we define the trajectory graph as a directed attributed graph $\mathcal{G}_{\mathcal{T}} = (V, E, \mathbf{A}_{\mathbf{v}}, \mathbf{A}_{\mathbf{e}})$ where V are the nodes, E the edges, and $\mathbf{A}_{\mathbf{v}}$ and $\mathbf{A}_{\mathbf{e}}$ are the node and edge attribute vectors respectively. These vectors of attributes serve the same purpose as the scalar weights commonly used in weighted directed graphs. To compare the weight of nodes and edges, we can use a vector similarity function as explained in Section 4.2.4.1 and Eq. 4.4. Each node is mapped to a vector of attributes that include: the *pose* ($\mathbf{x}$), *time of arrival* and optionally, the observed *altitude* (using barometer data, if available). All the node attributes are normalized relative to the initial node's observed values $\mathbf{A}_{\mathbf{v}0}$.

Edges contain the following vector of attributes $\mathbf{A}_{\mathbf{e}}$ that are calculated from corresponding node attributes upon edge creation: *driving duration*, *edge-length*, *edge orientation* and optionally, *edge inclination*. In fact, each attribute has a distribution that we assume to be Gaussian and hence conveniently represented by 2 parameters: the mean and variance.

The first node of $\mathcal{G}_{\mathcal{T}}$ represents the initial pose $\mathbf{x}_0$ i.e. the position where the vehicle is detected to have arrived for parking as explained in Section 4.2.2.1. Whenever we obtain a new pose $\mathbf{x}_{i+1}$, we look for major changes in their attributes across a sliding window of size W_t . Specifically, whenever we encounter any major change in any of the properties listed below, we insert a new node and associated edge(s) with the corresponding attributes.

- turn angle change $\delta\theta \geq \theta_{thres}$
- inclination angle $\delta\varphi \geq \varphi_{thres}$
- driving duration $\delta t \geq t_{thres}$

where θ_{thres} , φ_{thres} and t_{thres} are the respective threshold values of turn angle, inclination angle and driving duration with typical values 30° , 15° , 20s that can be tuned if required.

Furthermore, to simplify the graph for efficient processing, we try to merge the nodes using a modified Douglas-Peucker algorithm [92] for the below cases in particular:

- nodes with total *accumulated* turn angle $\Sigma\delta\theta > 90^\circ$ and located within a Euclidean distance d_{thres} (e.g. 50m) and traversed within a duration of W_t seconds
- nodes with *nearly straight* angles i.e. $\Sigma\delta\theta < \theta_{straight}$ (e.g. 15°) within a Euclidean distance d_{thres}

After ParkLoc has obtained the trajectory graph, it is stored into a Trajectory Graphs database and used for online localization (Section 4.2.4) or offline mapping (Section 4.2.5).

4.2.4 Graph Matching and Localization

In this section, we describe our localization approach by matching a trajectory graph $\mathcal{G}_{\mathcal{T}}$ against a map graph $\mathbf{M}$. Two representative examples of map graph $\mathbf{M}$ for real-life parking garages are illustrated in Fig. 4.3.

Figure 4.3: Examples of map graph super-imposed on floor plan: (a) CP_4 , a large shopping mall parking garage (b) CP_2 , a small indoor parking garage in a university campus. We refer to Section 4.3.1 for details on the test sites.

4.2.4.1 Graph - Sub-Graph Morphism Discovery Algorithm

To discover a sub-graph $\mathcal{G}_{\mathcal{T}}^*$ of map $\mathbf{M}$ that best matches the trajectory $\mathcal{G}_{\mathcal{T}}$, we propose to employ a modified version of the VF2 Algorithm [62] that adopts a Depth-First Search (DFS) approach with backtracking and uses a search tree to store and track the partial sub-graphs discovered as the algorithm proceeds. We adapt VF2 to find *semantic* feasibility of finding a match and to calculate *spatio-temporal* similarity between the candidate nodes/edges of the $\mathcal{G}_{\mathcal{T}}$ trajectory graph against those of $\mathbf{M}$. We emphasize that standard VF2 is designed for *exact* graph matching, which is not always possible in ParkLoc due to the noise-induced deformities in our trajectory graph. Therefore, we modify VF2 to try for *approximate* matching by considering the *feasible* nearest neighbors of candidate nodes. Fig. 4.4 illustrates the working of VF2 using an example of map and trajectory graph.

Each step of VF2 tries to find a mapping $\mathcal{M}$ between two graphs G and H . $\mathcal{M}$ is a set of pairs (n, m) with $n \in G$ and $m \in H$. It incorporates a state-space representation (similar to a search tree), using an intermediate state s of a recursive matching process $\text{Match}(s)$ as outlined in the Algorithm 2. Each state s corresponds to a *partial* solution $\mathcal{M}(s)$, which is only a subset of $\mathcal{M}$. The algorithm tries to identify the subset of valid and feasible states that meet several rules or consistency conditions[62] representing their *syntactic* and *semantic* feasibility. If a particular state is found infeasible, VF2 backtracks (abandons the state and its successors) and proceeds to the next state. This continues until a state that meets the feasibility requirements is found.

Syntactic (or structural) feasibility refers to whether the structural properties of the sub-graphs under $\mathcal{M}(s)$ are preserved, i.e., the edge connectivity between the currently compared nodes and their corresponding predecessors and successors, and whether some look-ahead rules [62] for pruning the search tree (and thereby reducing the search space) are satisfied. *Semantic* feasibility refers to whether the two nodes being compared have similar properties or not. In the context of ParkLoc, this gets extended to a *spatio-temporal* feasibility check, which can be represented by two sub-questions: i) do the two nodes have an approximately similar location in physical space? and ii) do they have a similar travel time from their corresponding predecessor or successor nodes as per the current state?

Figure 4.4: Illustration of our modified VF2 algorithm for graph matching: (i) map graph with entry at node a and exit at node h; orange line indicates the actual trajectory path, (ii) observed noisy trajectory graph, (iii) trajectory graph with virtual edges for approximate matching, (iv) resulting matched trajectory graph with a virtual edge chosen over real edge, (v) resulting matched sub-graph of the map, (vi) DFS search-tree and table that illustrates the evolution of the state-space as per VF2; it eventually chooses state s_6 (green) representing sub-graph $a-b-c-f$ over s_5 (red) representing $a-b-c-d$ as the final match, since s_6 satisfies all the syntactical, semantic and spatio-temporal feasibility conditions while s_5 is spatio-temporally infeasible.

In other words, ParkLoc extends the *semantic* feasibility checks on nodes and edges of $\mathbf{M}$ and $\mathcal{G}_{\mathcal{T}}$ so as to compare their *spatio-temporal* similarity and their temporal sequence as followed in the trajectory. To determine whether the candidate pairs identified by VF2 are similar or not, we compute pairwise Euclidean distances of the spatio-temporal properties of the nodes and edges, as listed in Section 4.2.3.2. To this end, we employ the spatio-temporal similarity metric based on radial basis function kernel as defined below, and we compare the edges (K_e) and nodes (K_v) based on their corresponding attribute vectors $\{\mathbf{A}_{\mathbf{e}_G}, \mathbf{A}_{\mathbf{e}_H}\}$ and $\{\mathbf{A}_{\mathbf{v}_G}, \mathbf{A}_{\mathbf{v}_H}\}$:

$$\begin{aligned} K_e(\mathbf{A}_{\mathbf{e}_G}, \mathbf{A}_{\mathbf{e}_H}) &= \exp\left(-\frac{\|\mathbf{A}_{\mathbf{e}_G} - \mathbf{A}_{\mathbf{e}_H}\|^2}{2\sigma_e^2}\right), \\ K_v(\mathbf{A}_{\mathbf{v}_G}, \mathbf{A}_{\mathbf{v}_H}) &= \exp\left(-\frac{\|\mathbf{A}_{\mathbf{v}_G} - \mathbf{A}_{\mathbf{v}_H}\|^2}{2\sigma_v^2}\right). \end{aligned} \quad (4.4)$$

In order to obtain better matching results with approximate graph matching, we perform several additional steps (refer lines 5-10 of Algorithm 2). In particular, we include the ζ -nearest neighbour nodes (typical value being $\zeta = 2$) of each edge being compared, as extra candidate node pairs to consider their similarity and determine a match. Whereas the standard VF2 algorithm considers only the direct node connections, we also inspect the *virtual* edges formed with their neighbors to see whether they at least match *approximately*. Since a more detailed explanation of VF2 is beyond the scope of this thesis, we refer interested readers to [62].

4.2.4.2 Locating the Vehicle

When VF2 is successful, we obtain the mappings set $\mathcal{M}$ from which we induce $\mathcal{G}_{\mathcal{T}}^*$, the sub-graph of $\mathbf{M}$ that best matches the trajectory $\mathcal{G}_{\mathcal{T}}$. We propose that the terminal edge e_n^* of $\mathcal{G}_{\mathcal{T}}^*$ is the *most likely* parking aisle link (edge) closest to the current vehicle position. Now, we use an online linear regression model to learn the relationship between properties of all the corresponding edges of $\mathcal{G}_{\mathcal{T}}$ and $\mathcal{G}_{\mathcal{T}}^*$ except $\langle e_n, e_n^* \rangle$. We use this regression model to *project* the latest trajectory edge e_n onto the corresponding map edge e_n^* which may have been only partially traveled, and thus estimate a fairly accurate location on this map edge. Based on the precision that ParkLoc obtains in a particular garage (as we shall ascertain through experimental evaluation in Section 4.3), we may localize the vehicle to a specific parking space, an aisle/driveway segment or a zone of a large parking garage.

Algorithm 2: Modified *VF2* Algorithm: **Match**(s)

Input: Intermediate state s with mappings between two graphs G and H ; for initial state s_0 , $\mathcal{M}(s_0) = \emptyset$

Output: $\mathcal{M}$, the mappings between the two graphs

```
1 if  $\mathcal{M}(s)$  covers all the nodes of  $H$  then
2   | return  $\mathcal{M}(s)$ 
3 else
4   | compute  $P(s)$ , the set of candidate pairs  $\langle g_i, h_i \rangle$  of nodes to include in  $\mathcal{M}(s)$ 
5   | get the set  $\eta_i$  of  $\zeta$ -nearest neighbours of  $h_i$ 
6   | foreach  $h_i$  in  $\eta_i$  do
7     |   if  $\langle g_i, h_i \rangle \notin P(s)$  then
8       |     | insert extra candidate pair  $\langle g_i, h_i \rangle$  to  $P(s)$ 
9       |   end
10  | end
11  | foreach  $p$  in  $P(s)$  do
12    |   if structurally, syntactically and spatio-temporally feasible to include  $p$ 
13      |     | in  $\mathcal{M}(s)$  then
14        |       | compute the state  $s'$ , by adding  $p$  to  $\mathcal{M}(s)$ 
15        |       | call Match( $s'$ )
16      |     | end
17    |   end
18 end
19   | restore data structures
20 end
```

4.2.5 Mapper

In this section, we explain the working of ParkLoc’s semi-supervised GraphSLAM-based mapping framework, i.e., how to learn the map graph $\mathbf{M}$ from sensed trajectory graphs.

4.2.5.1 GraphSLAM Overview

The map generation problem can be addressed using SLAM algorithms such as GraphSLAM [53, 55] developed by the robotics community. In general, the full SLAM problem deals with 2 sub-problems: (1) building (learning) a map graph and (2) localizing using the latest map, and we are particularly interested in the former.

GraphSLAM operates on a *pose graph*, consisting of vehicle *pose* nodes and edges connecting them. Each *pose* node consists of a 2D position and orientation *relative* to the initial pose $\mathbf{x}_0$. Each edge of this graph represents the spatial constraint between a pair of nodes, with a probability distribution encoded as its information matrix. For e.g., a *motion constraint* is generated from inertial sensor measurements when the vehicle moves from node $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$, and represented by the relative 2D translation and orientation change. The pose graph also includes nodes and edges (constraints) for any *landmarks* observed along the vehicle’s path, relative to its pose. Given the poses, landmarks and their constraints, GraphSLAM helps us find their optimal spatial configuration i.e., the *mostly likely map*. During each iteration of GraphSLAM algorithm, it can update the pose graph as it tries to minimize the errors. In short, GraphSLAM offers a *least squares* error minimization approach to SLAM, with a graph-based formulation.

ParkLoc sources its pose graph primarily from the trajectory graph explained in Section 4.2.3.2; landmark nodes are obtained separately as explained below. Fig. 4.5 provides an illustration of the GraphSLAM concepts in the ParkLoc context, by using a subset of the pose graph recorded at a parking garage (*CP4*, see Fig. 4.3a), relevant landmarks and the corresponding map estimated using GraphSLAM.

4.2.5.2 Acquiring Landmarks: Why, and How

When errors build up due to the noise in our vehicle motion model, landmarks help GraphSLAM pull the poses closer to the true values. In contrast to robotic systems that utilize exteroceptive sensors to obtain position estimates of landmarks along their trajectories, a smartphone-only ParkLoc may seem to be handicapped in acquiring landmarks. Fortunately, ParkLoc is able to acquire two kinds of landmarks: *global* and *intrinsic*. To bootstrap our ParkLoc Mapper, we follow a semi-supervised approach by taking advantage of a few *global* landmark nodes λ_{glm} as seeds¹ that have predefined indoor locations in the garage. Each seed is represented by an edge containing a relative transformation from $\mathbf{x}_0$ directly to its true location, denoted by the black dashed edges in Fig. 4.5.

Nevertheless, there are some *intrinsic* landmarks λ_{intr} that are inherent to the parking garage structure that ParkLoc can recognize and rely on for *data association* and to achieve *loop closure* (during multiple visits), even in our non-instrumented environments. Specifically, we consider the major turning points in a garage and their sequences such as sharp 90° turns, 180° turns, a cluster of two closely located 45° turning points, or a sequence of *<right, left, left<* turns within a short driving duration and small odometric distance (e.g., the orange-colored landmark shown in Fig. 4.5). Fortunately, ParkLoc can obtain a sufficient number of such intrinsic landmarks from the trajectory graphs in most of the indoor parking garages, as we can discern from the maps in Fig. 4.3. Whenever an *intrinsic* landmark with a fairly high certainty is detected during vehicle motion, ParkLoc inserts a zero-length edge into the pose graph at the current pose.

4.2.5.3 Theoretical Formulation and Error Minimization

Now, we briefly outline the theoretical details of GraphSLAM. Formally, we model the SLAM problem as an *overdetermined* system with more observations (constraints or edges obtained from measurements) than states (*pose* nodes), and try to determine the most likely configuration of states given the constraints i.e. the *most likely map*, using least squares optimization. Our objective is to minimize the sum of squares of

¹Seeds are provided by users, e.g., pushing a button on the smartphone app when the vehicle arrives at the corresponding locations.

Figure 4.5: Illustration of GraphSLAM concepts. Map edges spatially constrain the red-colored pose nodes $\mathbf{x}_i$ based on noisy odometry with high uncertainty (illustrated via large circles); however, these nodes get iteratively optimized to $\mathbf{x}_i^*$ as the algorithm proceeds. Global landmark edges directly connect the origin $\mathbf{x}_0$ to yellow-colored *seeds* (λ_{glm}) with low uncertainty. Orange-colored *intrinsic* landmarks (λ_{intr}) are automatically recognized from the planar structure and help optimize the map. Note that circles illustrating the uncertainty do not represent their actual distribution.

error $\mathbf{e}_{ij}$ between measured value of a constraint and its predicted value based on the current pose graph.

Consider the full set of nodes $\chi = (\mathbf{x}, \lambda_{glm}, \lambda_{intr})$, which includes all the nodes for which the pose graph contains defined constraints i.e., all vehicle poses and landmark nodes. Given χ , GraphSLAM aims at obtaining χ^* , the optimal configuration of nodes. To this end, we minimize the negative log likelihood $\mathcal{L}$ of all nodes χ over the set B of edges, where each edge has a mean error $\mathbf{e}_{ij}$ and an information matrix $\mathbf{\Omega}_{ij}$ representing its uncertainty:

$$\mathcal{L}(\chi) = \sum_{(i,j) \in B} \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij} \quad (4.5)$$

$$\chi^* = \arg \min_{\chi} \mathcal{L}(\chi) \quad (4.6)$$

To solve Eq. 4.6, we adopt the Gauss-Newton algorithm for iterative local linearization, given a good estimate of the initial pose $\mathbf{x}_0$. This is relatively efficient since it does not require the computation of second derivatives. We assume that $\mathbf{\Omega}_{ij}$ is always diagonal, i.e. only variances are considered. Constraints for λ_{glm} have low uncertainty,

whereas those representing motion ($\mathbf{x}$) and λ_{intr} have much higher uncertainty. In Fig. 4.5, these are differentiated by the small yellow circles for λ_{glm} and larger red circles for $\mathbf{x}$.

Initially, we assume that all constraints for $\mathbf{x}$ and λ_{glm} are *satisfied* whereas those for λ_{intr} and loop closure are initially *violated*. However, in the subsequent iterations, the algorithm progressively optimizes the poses. At each step, a suboptimal set of updated nodes $\tilde{\chi}^*$ is computed given the current constraints and uncertainty; all the constraints are then reevaluated based on $\tilde{\chi}^*$. This procedure is repeated (with iteratively updated pose graph) until eventual convergence, when it yields the desired optimal set of nodes $\chi^* \cong \mathbf{M}$. We also note that due to the topological approach we adopt by using a sparse trajectory graph with simplified nodes (as described in Section 4.2.3.2), we are able to work with a significantly *reduced* SLAM state space, which can otherwise become quite large as in typical robotic systems.

4.3 Empirical Evaluation

In this section, we describe the evaluation procedure and setup for data collection and experiments using our ParkLoc prototype and examine the results to gauge the performance.

4.3.1 Experimental Setup

In order to ascertain the real-life performance of the methodology proposed in ParkLoc, we have implemented ParkLoc as an Android-based application running on a driver’s smartphone along with a back-end mapper service. While the sensor data collection, processing, maneuver points detection and trajectory graph generation are performed by the smartphone app, the GraphSLAM-based mapping algorithms are performed in the server using the data reported by the ParkLoc app, which in turn receives the up-to-date map graph back from the server.

For a comprehensive evaluation, we conduct experiments in 4 different parking facilities that vary in size, structure and occupancy patterns as described in Table 4.1. Experiments were performed using 4 different phones: Google (LG) Nexus 5, HTC One

M8, Samsung Galaxy S4 and Samsung Galaxy Note 4. All of these possess accelerometer and gyroscope sensors. We collected more than 100 traces from 6 different days using 4 different drivers and 3 different cars (Toyota Wish MPV, Toyota Prius C hybrid hatchback, and BMW i3 EV). In general, we performed a simple procedure: from a predefined starting point, drive to a few specified parking spaces, drive back to the starting point, and repeat. While the phone attitude and placement is arbitrary as per the choice of each driver in our experiments, in general the phone was not hand-held and remained fixed during driving.

Table 4.1: Description of parking facilities used for the empirical evaluation of ParkLoc

Name	No. of spaces	Trajectory length (m)	Map area (m ²)	Type
<i>CP1</i>	150	270	3540	University, outdoor, surface level
<i>CP2</i>	80	560	5930	University, indoor, underground
<i>CP3</i>	250	300	3110	Shopping mall, indoor, underground
<i>CP4</i>	750	2025	24200	Shopping mall, indoor, underground

4.3.2 Data Pre-processing

After collecting the data, we annotated it with the ground truth location information for evaluation purpose. Ideally, GraphSLAM does not have any requirement for labeled data, and the ground truth is used only for evaluation. We used ground truth accurate to about 1m. In case of *CP1*, this procedure was easy to perform due to the strong GPS signal reception and required slight manual corrections only for a few instances where the number of GPS fix satellites was < 7 . For *CP2*, *CP3* and *CP4*, we adopted different methods such as super-imposing a 2D floor plan on to high-resolution satellite imagery and physical distance measurements followed by manual corrections, as they did not receive GPS signals indoor. To apply manual corrections, we identified location-invariant structural artifacts that are present both above and below ground such as pillars, elevators or staircases (which are continuous across floors and hence visible on satellite maps) and exploited them as markers along with our measurements. For each of these parking facilities, we selected some strategically important indoor locations as our seed landmarks. To evaluate the graph isomorphism-based localization, we manually

prepared topological map graphs that are easily generated from their corresponding floor plans (we could complete this process within few minutes, using a web-based mapping tool).

4.3.3 Graph-based Localization Accuracy

In this section, we explain how the graph-based online localization algorithm of ParkLoc performs given a map graph of the parking garage. The map graph can be either supplied beforehand (of high quality as it is manually corrected and verified based on ground truth) or generated using the ParkLoc Mapper. Therefore, we evaluate the localization performance under both cases.

4.3.3.1 Localization Performance

The clear benefit of the graph-based approach can be understood when we compare the localization performance to a much simpler dead-reckoning-based approach by directly using our optimized vehicular motion model presented in Section 4.2.3.1. From Fig. 4.6(b,d), we can see that median dead-reckoning errors can be around 25m and can go even to about 70m in a worse case. This is most likely because the high sensor and process noises accumulate heavily causing the position estimates to drift further away from their true values, especially while driving long straight paths. In contrast, the graph-based ParkLoc localization algorithm performs very well with most (90th percentile) real-time tracking errors within 0-16m and parking localization errors within 0-6m when a high quality map graph is used. Using a slightly inferior uncorrected map graph auto-generated by ParkLoc mapper for map quality evaluation under same settings, ParkLoc can still obtain a useful performance with median errors of 8m and 6m respectively for real-time tracking and parking localization. However, the worst case errors are higher and go up to about 45m and 30m respectively due to the errors within the map graph generated by ParkLoc mapper. We will discuss the map generation performance and map quality aspects later in Section 4.3.4 using specific examples as in Fig. 4.13.

As we can discern from Fig. 4.7a and 4.7b, the graph-based localization and tracking algorithm performs well with optimal values of the dead-reckoning calibration parameter k . We obtain the best performance when $k = 0.35$, since we obtain median tracking

Figure 4.6: (a,c) Cumulative Distribution Function (CDF) plot of real-time tracking and parking localization error of ParkLoc vs. a Dead Reckoning approach, both using calibration parameter $k=0.35$ (b,d) Histogram showing the distribution of ParkLoc errors. Sub-figures (a,b) indicate the performance obtained with true, manually corrected map graphs, whereas (c,d) indicate the corresponding performance with maps auto-generated by ParkLoc mapper at the same parking garages.

Figure 4.7: CDF plot of real-time tracking error for different values of the calibration parameter k , when evaluated with (a) the true, manually corrected map graph and (b) auto-generated map graph.

error of 4.8m (within 2 parking spaces) and error values are within 16m (about 5-6 parking spaces) for 90% cases using a high-quality map graph. In our application case where the driver wishes to use ParkLoc to find where the vehicle was parked, this error is tolerable and fits within the range of a wireless remote key. In fact, only in very few cases, we saw the real worst case performance in the range 30-40 m (about 10-12 parking spaces); even this is only a short distance that is easily traversed by walking.

Fig. 4.8 shows that the latency expected during real-time tracking as well as determining the location of a parked vehicle are within the practical ranges of (0.1s, 0.35s) and (0.2s, 0.4s) respectively. The optimal latency is obtained when the online trajectory graph is cutoff into the latest 4 nodes, which is a reasonably good approximation. From Fig. 4.9, we can also observe that the lowest errors are obtained when the vehicle is close to a graph node. This is equally applicable for both manually corrected (Fig. 4.9a) as well as auto-generated map graphs (Fig. 4.9b). The mean localization errors tend to peak around or slightly higher than 10m when the vehicle is located around the middle of a matched graph edge. This is especially noticeable in case of long straight aisles. Fig. 4.9 also indicates different variability (*interquartile range* or IQR) in localization errors as the vehicle moves along the edge farther from the graph node. The larger IQR at distances 20 and 40m attribute to the relatively lower sample sizes of observations made at these distances within our evaluation environments and

Figure 4.8: Latency for (a,c) real-time tracking and (b,d) parking localization under various trajectory graph cut-off (subgraph) sizes. Sub-figures (a,b) uses the true, manually corrected map graphs whereas (c,d) use auto-generated map graphs.

Figure 4.9: Localization error vs minimum distance from end points of the localized map graph edge for (a) true manually corrected map and (b) auto-generated map. ParkLoc seems to work best when vehicle is in proximity of a map graph node.

Figure 4.10: CDF plot of real-time tracking error for different values of the trajectory subgraph (cut-off) sizes, using (a) the true, manually corrected map graph and (b) auto-generated map graph.

may not be generalized to other settings.

4.3.3.2 Tuning the Optimal Trajectory Subgraph Size

As the vehicle arrives and navigates into the parking garage, the online trajectory grapher trims the trajectory information and stores only the most recent trajectory information to localize itself. We use this evaluation to obtain the ideal value of size of the trajectory subgraph H . As we can discern from Fig. 4.10a and Fig. 4.10b, we consistently obtain the lowest localization error with a trajectory subgraph of 4 recent nodes (i.e., $\|\mathcal{G}_T\| = 4$). This is found to be empirically true for both high-quality map graphs as well as auto-generated map graphs. Increasing the size, can cause the VF2 algorithm to both consume more time as well as make wrong estimates of the current graph edge.

4.3.3.3 Tuning the Sensing Window Size W_g

A purpose of the evaluation is also to determine the optimal window size W_g , that can provide us the best accuracy while guaranteeing an efficient operation. Fig. 4.11 shows that the lowest errors are obtained when W_g is 3 or 3.5s and the smallest latency at 2.5 and 3s. This is generally found to be consistent when localizing using both a high-

Figure 4.11: Choosing an optimal window size W_g for the vehicle motion model, that provides the best localization error and low latency, for (a) true manually corrected map and (b) auto-generated map.

quality map graph (see Fig. 4.11a) or auto-generated map graph from ParkLoc mapper (see Fig. 4.11b), a noticeable exception being the slightly lower error at 4s in the latter case. The latency increases with smaller window sizes, as this increases the number of motion model steps and thereby the graph size, resulting in longer computations. Considering the above, we choose to operate with $W_g = 3s$.

4.3.3.4 Performance Comparison with Related Works

To clearly appraise the contributions of ParkLoc, we study how the localization performance stands in comparison to the other related proposals. To this end, we compare the performance of ParkLoc with that of VeLoc [15] and its extension VeTrack [16]. To enable an effective comparison, we employ and evaluate the VeTrack' prototype, essentially a re-implementation of VeTrack that we have developed by closely following the literature, within our experimental environments. Our high-quality map graphs serve as the road skeleton models for VeTrack'. In addition to the above, we also refer to the literature and reproduce the original performance measurements reported by

Figure 4.12: Comparison of real-time tracking and parking localization performance between ParkLoc and related works.

VeLoc and VeTrack in their native deployments, after translating them to our context as described below.

VeLoc reports its performance in meters. However, VeTrack uses a different error metric, i.e., the number of parking spaces. To achieve a consistent comparison, we convert this to meters, assuming that the error is measured along the aisles (ParkLoc measures error along the graph). Therefore, we can multiply the number of spaces with the width of a rectangular parking space, that we choose as 2.7m after consulting the civil construction manuals of different countries. Since both VeLoc and VeTrack report errors from 3 separate garages, we average these and compare with the errors produced by ParkLoc and VeTrack' in 3 garages of comparable size and structure.

From Fig. 4.12, we can discern that ParkLoc offers significantly better performance than VeTrack' for both tracking as well as parking localization. For our main motivating application case of parking localization, ParkLoc offers the median and 90th-percentile errors of 2m and 6m respectively, whereas VeTrack' incurs larger errors of 6m and 15m respectively under the same evaluation conditions. In case of real-time tracking of the vehicle, ParkLoc still offers higher performance of 4.8m and 15.7m in comparison to the corresponding errors of 7.6m and 20.4m by VeTrack'. Clearly, despite being a light-weight approach, ParkLoc offers a substantially useful performance. In fact, the performance difference between VeTrack' and its native version can be partially attributed to its low robustness given complicated machine learning procedures requiring parameter fine-tuning.

Even when directly compared to the performance reported by VeLoc and VeTrack in their native deployments, ParkLoc can offer similar or better performance. Although

worse case parking localization errors are slightly more than VeLoc, the real-time tracking errors are slightly lower or similar to VeTrack (both median and 90-percentile) although worse case errors seem to be much higher. This performance difference could be because of any unknown differences in the parking facilities and/or evaluation procedure. Another possible reason could be that while ParkLoc measures the error along the graph, other approaches may instead use Euclidean distance. We also note that ParkLoc does not include detection of speed bump landmarks (which are unfortunately not common inside the parking garages that we evaluated), in contrast to VeTrack that uses supervised machine learning to detect them; therefore, ParkLoc tracking errors may be larger on long straight paths, considering the noisy odometry and unavailability of landmarks or turns. However, ParkLoc offers the advantage that it does not require any substantial training or data collection for localization given a map graph of the garage and this can result in faster deployments. In addition, ParkLoc can automatically learn the map given only some seed nodes.

4.3.4 Mapping Performance

As we have already seen under Section 4.3.3, ParkLoc can localize reasonably well using the map graphs automatically generated by the Mapper. However, the localization accuracy obtained is not as good as a high quality map graph prepared from floor plans and manually corrected for precision. To better understand and quantify the differences between these map graphs, it is important to perform a holistic evaluation of the mapper performance in terms of generated map quality. To this end, we consider the 4 parking facilities CP1-4 that have moderate complexity, realistic sizes and good diversity in terms of their map graphs.

Fig. 4.13a-d presents the corresponding results from these 4 cases and shows how the GraphSLAM-based mapper performs in our restricted sensing environment from mobile sensed vehicle trajectories. Whereas the learnt map is not an exact replica of the true map, it is topologically very similar in all these evaluated cases and this is more than sufficient for our VF2-based inexact graph matching method to provide a high localization accuracy. Whereas obtaining loop closure is quite difficult (or almost impossible in the presence of high noise) using only vehicular dead reckoning, our

Figure 4.13: Mapping performance at 4 parking facilities $CP1$, $CP2$, $CP3$ and $CP4$. (a-d) visualization of the learnt maps (red dotted lines) superimposed upon the true maps (blue lines), GPS trajectory if available (green lines), seeds and intrinsic landmarks (yellow and orange circles) (e) evaluation of mapping error vs number of seed nodes at different threshold turn angles θ_{thres} .

semi-supervised approach helps us achieve loop closure while preserving the topological connectivity and structure.

In order to ascertain the mapping performance, we establish the following error metric that can compare the positions on the learnt map to their true values (that we obtained during our pre-processing step in Section 4.3.2). For a true map $\mathbf{x}_{1:T}$ with T pre-determined poses, we compute the quality of the generated map $\mathbf{x}_{1:T}^*$ as the root-mean square error (RMSE) from their pairwise Euclidean distances.

$$\Upsilon(\mathbf{x}_{1:T}, \mathbf{x}_{1:T}^*) = \sqrt{\frac{\sum_{t=1}^T (\mathbf{x}_t - \mathbf{x}_t^*)^2}{T}} \quad (4.7)$$

Using this metric, we now analyze the mapping performance and understand how the number of seed nodes and system parameters affect the quality of the generated map. From Fig. 4.13e, we can observe that, depending on the structure of the garage, quality of the map improves significantly when we introduce a few seed nodes. For CP1, we get acceptable performance with RMSE of about 2.5m with 4 seed nodes. We see similar results for CP2 (4 seeds) and CP3 (5 seeds). However, we can also discern that any additional seed nodes do not improve the performance much, since the resulting map is already topologically very similar to the true map, even though not exactly equal. Note that in case of larger and more intricate parking garages such as CP4, proportionally more seeds are required to generate a map of high quality with low RMSE.

The evaluation also throws light on the influence of system parameters such as the threshold turn angle (θ_{thres}) used to generate vehicle motion-based pose nodes in the trajectory graph that are further optimized by GraphSLAM. Different values of θ_{thres} may yield slightly different performance for different garages. For CP1, we obtain similar performance for θ_{thres} values 10°, 30° or 60°. For the other 3 garages, we obtain different values of RMSE with no seed nodes. However, with a sufficiently high number of seed nodes, θ_{thres} values do not make much impact as we obtain low RMSE values. After a careful consideration of all the evaluation cases, we choose $\theta_{thres} = 10^\circ$ as a typical value that can work well in general. However, this can be further tuned based on individual requirements.

Fig. 4.14 clearly shows the influence of landmarks on the mapping performance. We note that the number of *intrinsic* landmarks λ_{intr} remains fixed for each parking

Figure 4.14: Impact of landmarks (seed or intrinsic) on Mapper performance. Figure shows the number of landmarks required to achieve a minimum map quality in terms of RMSE (1, 2.5, 5 or 7.5 m) for CP1-4. Number of intrinsic landmarks detected remains fixed for each parking garage and depends on its structure. For larger sized garages and/or those with a small number of uniquely distinguishable intrinsic landmarks, more seeds are required to achieve good performance.

garage in general, depending on the structure and layout of the navigable indoor paths. If the number of λ_{intr} is large enough in proportion to the physical area of the garage, we can achieve a good map quality with less seeds. We can observe this in case of CP1-3, wherein an RMSE of at least 2.5m is achievable with 3-4 seeds. However, parking garages such as CP4, that are much larger but have mostly 90° right turns, may not have a sufficient number of intrinsic landmarks that can uniquely represent the indoor locations by using their natural sequence of turns alone. To handle such cases and attain a decent map quality without introducing the overhead of any additional computationally intensive landmark detection algorithms, we will require the user to insert more seeds.

Although several proposals exist that perform general indoor mapping using smartphones, they are not directly applicable to the needs of our ParkLoc localizer since they are mostly designed for pedestrian localization using sensors (such as WiFi) that ParkLoc cannot utilize. To address the specific problem of mapping indoor parking garages for vehicular localization using only smartphones, Zhou et al. [93] has recently proposed a few specially crafted algorithms, including sequential merging/clustering of

points, as well as matching of partial line segments from several driver trajectories. As custom-made solutions, they provide high accuracy although they may require more elaborate preprocessing. In contrast, the ParkLoc mapper offers useful performance based on a light-weight and more general SLAM-based solution: it directly produces the map as a simple and optimal pose graph structure, which graph-based localization methods (e.g., ParkLoc) may leverage as we have seen through the experimental evaluation under Section 4.3.3. In addition, the GraphSLAM framework that we adopt has a strong theoretical basis with nice convergence properties [55] when given sufficient sensor data inputs from vehicle motion and environment perception. Therefore, the ParkLoc mapper is more scalable towards integrating any future enhancements including additional sensors or data sources if and when they become available.

4.4 Discussion

In this section, we discuss some key considerations, special cases and challenges to be deliberated with regard to practical deployments of ParkLoc.

4.4.1 Additional infrastructural support

Although ParkLoc is designed to work without relying on infrastructure, a practical deployment can always take advantage of any relatively lower cost infrastructure. For example, installing a few BLE beacons at some specific locations to serve as seeds (global landmarks) could bring out a significant improvement in the mapping performance as well as the localization accuracy. This could be beneficial especially in long straight line paths without any good potential of detecting intrinsic landmarks. Similarly, ParkLoc could also take advantage of WiFi access points, wherever available, to get landmark information.

4.4.2 Garage constraints

ParkLoc experiments were conducted primarily in parking garages with mostly one-directional travel lanes, except for some small areas near entry/exit. However, in case of garages where bidirectional traffic is allowed, the GraphSLAM based mapper is expected to perform better. This is particularly applicable to zones with very long

straight line segments (edges) with parallel incoming cross roads all incident at the same angle, such as in case of garage CP4 illustrated earlier under Fig. 4.3 and Fig. 4.13d. This is because the GraphSLAM algorithm can perform better data association in such cases and converge faster with a much lower number of seeds.

4.4.3 Multi-floor localization

For simplicity, the current ParkLoc experimental evaluation considered the vehicular localization to be performed only within a single floor of a garage. However, this can be easily extended to multi-floor localization use cases. As briefly mentioned earlier, the use of smartphones equipped with barometer will help us obtain the relative height (elevation) w.r.t. the garage entry point and the road gradient (slope) of each edge. The proposed framework already considers such information on height and elevation as optional attributes of the graph nodes and edges, and these can be readily used to perform localization if and when available, by means of the spatio-temporal similarity-based graph matching procedure. Optionally, the system can also incorporate the floor change detector (FCD) module, that we have separately proposed in Chapter 3, to first recognize the correct floor the vehicle is in, and then perform intra-floor localization using the corresponding map graph for that floor.

4.5 Summary

In this chapter, we have proposed a graph-based method named ParkLoc that uses only commodity smartphones for the purpose of achieving light-weight indoor localization of vehicles in a parking garage. ParkLoc leverages the natural sparse topological graph structure of vehicle trajectories in parking garages that can be easily discovered using only inertial sensors. We propose a novel localization approach by extending state of the art graph matching methods. Given a high-quality map graph for the garage, ParkLoc can offer very useful results with a median error of 4.8m and 2m for real-time tracking and parking localization respectively, with no additional training effort per garage. Furthermore, we demonstrate how ParkLoc can learn the garage map graph from driver trajectories, using graph-based SLAM methods.

Chapter 5

Conclusion

Mobile sensing and crowdsensing have recently emerged as powerful tools to enable different applications and services; on the other hand, several machine learning and pattern recognition methods are now viable enough to run on mobile devices. In this thesis, we aim to encourage greater adoption of these tools to help solve mobility problems. To this end, we have explored the potential of applying machine learning to mobile sensor data to address the first/last-furlong problems in private vehicular mobility, such as to find an available parking space, or to localize/track a (parked) vehicle in large indoor parking garages in the absence of supporting infrastructure such as GPS or WiFi. In Chapter 3, we have proposed a promising approach towards parking occupancy inference. Furthermore, we also set out to answer the question of whether we can achieve scalable and sustainable crowdsensing systems without repeating the expensive data gathering steps. In Chapter 4, we have proposed a light-weight graph-based approach to directly address the problem of indoor vehicular localization/tracking and mapping, which is relevant for both last-furlong and first-furlong situations.

We conclude this thesis by summarizing our contributions and discussing some future directions in the following sections 5.1 and 5.2 respectively.

5.1 Research Contributions

In Chapter 3, based on the observation that indoor parking garages can lack real-time occupancy information, we have proposed ParkGauge+, a mobile crowdsensing system

for estimating the occupancy of indoor parking garages. The existing proposals focus on outdoor parking and cannot be applied here, since directly counting indoor occupied spaces is infeasible due to the unavailability of precise localization infrastructure that these proposals expect. Instead, our design of ParkGauge+ innovates in estimating the parking occupancy by establishing the correlation with temporal parking characteristics (that we compute from inertial sensor data) such as the total time a vehicle spends in finding a parking space (**time-to-park**) or waits in a queue (**time-in-queuing**) both at the entrance while approaching the garage and at indoors. The key advantage of this method is that this is less sensitive to the crowdsensing penetration ratio and can operate well with very few participants per garage. On the other hand, this method per se introduces no need to precisely track a vehicle's indoor location to infer the parking occupancy. In particular, our work has made the following important contributions:

- a novel idea of using **time-to-park** as an indicator to infer parking occupancy, by exploiting their strong correlation which is both intuitively and empirically valid.
- a hierarchical mobile sensing architecture to automatically discover the parking process at different levels of detail, from raw inertial sensor data to driving states (e.g., **turning,braking**) to driving contexts (e.g., **cruising, parked**), and produce features to train machine learning algorithms (such as SVR-PUK) leading to robust occupancy inference.
- a smart mobile sensing methodology that describes how the temporal evolution of various states and sub-states of a sensing system can be hierarchically fused to infer a *macro*-state of its environment; in particular, how driving states and contexts can be availed to compute parking characteristics of a garage (e.g., **time-to-park**) and used as implicit indicators of the occupancy experienced in the garage.
- the novel introduction of transfer learning methods into crowdsensing (as we call *transfer crowdsensing*) to ensure scalable deployment of such systems at new, unseen venues while minimizing the expensive data collection and labeling efforts.
- exhaustive experiments at large, commercial parking garages to demonstrate how ParkGauge+ performs with both supervised learning on rich source dataset as well as with transfer learning when given only small target data for new garages.

In Chapter 4, we have proposed ParkLoc, a novel light-weight graph-based approach for localizing vehicles within indoor parking garages, which is infrastructure-free and consumes low setup effort. This will help users to easily find their parked vehicles, and also enhance parking occupancy inference systems such as ParkGauge+ with additional localization capabilities. In particular, we make the following four main contributions:

- A systematic approach for localizing and tracking a vehicle inside a large parking garage, based on the novel idea of using spatio-temporal characteristics of (a sequence of) turn maneuvers as the approximate signature of indoor locations.
- A method for constructing a motion trajectory of the vehicle as a planar graph assembled from commodity smartphone sensor data, which is used for online localization by finding an *inexact* subgraph isomorphism within a map graph.
- A semi-supervised GraphSLAM-based map learning approach to approximate the map of a garage as a planar graph that aids in online localization.
- An exhaustive empirical evaluation in different parking garages of diverse shape and structure, that clearly demonstrates the usefulness and generality of the proposed methods for indoor vehicular localization in parking garages.

5.2 Future Directions

Although ParkGauge+ is able to perform well with reasonably low errors in our experimental evaluation, there are several areas it could be improved. The supervised machine learning deployments offer higher accuracy than transfer learning-based variants, however this comes at the expense of some efforts for data collection, training and tuning. Our evaluation using inductive transfer learning methods produced promising results that demonstrate the merit in this approach; however, we believe that the re-usability of each source dataset does depend on the domain and distribution, and therefore the generality can only be ascertained with big data. Due to the insufficient resources at our disposal, we could not make large-scale trials involving a crowd of hundreds or thousands of users which would have been ideal. Therefore, our empirical dataset has so far been quite limited and the potential of our proposed methods with big data remains unknown. To bridge this gap in near future, we plan to make larger scale deployments

of ParkGauge+ using both traditional and transfer learning approaches and study the impact of diversity in source-target data distributions. Furthermore, since our current transfer learning method still requires some small training data for the target garages, we plan to explore *zero-shot*, *one-shot* and *few-shot* learning methods using zero, one or only few training examples. Although we have carefully engineered our ParkGauge+ prototype to estimate parking characteristics and infer occupancy from use low-power inertial sensors, our hierarchical mobile sensing/crowdsensing methodology may not be restricted to parking or mobility problems alone; it can potentially serve many other purposes beyond occupancy inference. In general, a similar architecture could be applied to other processes that can be modeled as sensor-derived states and sub-states that evolve temporally resulting in a *macro*-state of the environment where the applicable sensors are placed.

The success of any localization scheme rests on the quality of the map. ParkLoc currently achieves the desired map quality using a semi-supervised learning approach. Due to the unavailability of GPS/WiFi signals in parking garages, non-usability of geo-magnetic field within vehicles for location fingerprinting, and the absence of exteroceptive (laser or visual ranging) sensors used by robotic vehicles, we currently require some seed landmark nodes to bootstrap the GraphSLAM-based mapper. However, we aim to find ways to relax this requirement and to get good mapping performance in cold-start situations with no prior knowledge.

In a full-fledged commercial deployment of ParkLoc, we can make the following additional improvements even though we considered them out of the scope of this thesis due to their dependence on infrastructure and/or training requirements:

- a) To improve the parking arrival detection (Section 4.2.2.1) and better disambiguate closely located garages in dense urban canyons, ParkLoc may incorporate WiFi-based outdoor localization [30] or even a supervised hands-free voice dialogue scheme that uses driver’s voice input to confirm the correct parking garage
- b) ParkLoc may automatically obtain seed landmarks using cheap infrastructure supports such as zone identifiers, QR codes, RFID or NFC-tagged zones, or WiFi/BLE RSSI measurements in zones with limited signal coverage, wherever they are available, and

- c) By applying machine learning and pattern recognition methods, ParkLoc can recognize more intrinsic landmarks such as entry/exit gates, ramps, waiting points, very-low-speed areas or even speed bumps, after performing the necessary training and tuning procedure.

Finally, we plan to extend the GraphSLAM framework to a full scale crowdsensing paradigm, to achieve joint map learning from various crowdsensed trajectories and improve the mapping performance.

References

- [1] YUANCHAO SHU, KANG G SHIN, TIAN HE, AND JIMING CHEN. **Last-mile navigation using smartphones**. In *Proc. of the 21st ACM MobiCom*, pages 512–524. ACM, 2015. 1
- [2] DONALD C SHOUP. **Cruising for Parking**. *Elsevier Transport Policy*, **13**(6):479–486, 2006. 2, 14
- [3] INRIX INC. **New INRIX Study Finds Parking is the Largest Cost of Driving** [online]. 2017. <http://web.archive.org/web/20180429220524/http://inrix.com/press-releases/cod-us>. 2
- [4] RICHARD ARNOTT. **Spatial competition between parking garages and downtown parking policy**. *Transport Policy*, **13**(6):458–469, 2006. 2
- [5] DANI SIMONS. **SFpark: San Francisco Knows How to Park It**. *ITDP Sustainable Transport*, (23), 2012. 2, 15
- [6] SUHAS MATHUR, TONG JIN, NIKHIL KASTURIRANGAN, JANANI CHANDRASEKARAN, WENZHI XUE, MARCO GRUTESER, AND WADE TRAPPE. **ParkNet: Drive-by Sensing of Road-Side Parking Statistics**. In *Proc. of the 8th ACM MobiSys*, pages 123–136, 2010. 2, 6, 14, 15
- [7] DANIEL AYALA, OURI WOLFSON, BO XU, BHASKAR DASGUPTA, AND JIE LIN. **Parking Slot Assignment Games**. In *Proc. of the 19th ACM GIS*, pages 299–308, 2011. 2
- [8] NICHOLAS D LANE, EMILIANO MILUZZO, HONG LU, DANIEL PEEBLES, TANZEEM CHOUDHURY, AND ANDREW T CAMPBELL. **A Survey of Mobile Phone Sensing**. *IEEE Comm.*, **48**(9):140–150, 2010. 3, 8, 16

-
- [9] TRISTA LIN, HERVÉ RIVANO, AND FRÉDÉRIC LE MOUËL. **A Survey of Smart Parking Solutions.** *IEEE Trans. on Intelligent Transportation Systems*, 2017. 3, 7, 8
 - [10] RAGHU GANTI, FAN YE, AND HUI LEI. **Mobile Crowdsensing: Current State and Future Challenges.** *IEEE Comm.*, 49(11):32–39, 2011. 3, 8, 16
 - [11] SARFRAZ NAWAZ, CHRISTOS EFSTRATIOU, AND CECILIA MASCOLO. **ParkSense: A Smartphone Based Sensing System for On-Street Parking.** In *Proc. of the 19th ACM MobiCom*, pages 75–86, 2013. 3, 7, 8, 16
 - [12] ANANDATIRTHA NANDUGUDI, TAEYEON KI, CARL NUESSELE, AND GEOFFREY CHALLEN. **PocketParker: Pocketsourcing Parking Lot Availability.** In *Proc. of the 16th ACM UbiComp*, pages 963–973, 2014. 3, 7, 16
 - [13] SAMULI HEMMINKI, PETTERI NURMI, AND SASU TARKOMA. **Accelerometer-based Transportation Mode Detection on Smartphones.** In *Proc. of 11th ACM SenSys*, page 13, 2013. 3, 9, 16
 - [14] KARTIK SANKARAN, MINHUI ZHU, XIANG FA GUO, AKKIEBBAL L ANANDA, MUN CHOON CHAN, AND LI-SHIUAN PEH. **Using Mobile Phone Barometer for Low-Power Transportation Context Detection.** In *Proc. of 12th ACM SenSys*, pages 191–205, 2014. 3, 31
 - [15] MINGMIN ZHAO, RUIPENG GAO, JIAXU ZHU, TAO YE, FAN YE, YIZHOU WANG, KAIGUI BIAN, GUOJIE LUO, AND MING ZHANG. **VeLoc: Finding Your Car in the Parking Lot.** In *Proc. of the 12th ACM SenSys*, pages 346–347, 2014. 4, 11, 54, 78
 - [16] MINGMIN ZHAO, TAO YE, RUIPENG GAO, FAN YE, YIZHOU WANG, AND GUOJIE LUO. **VeTrack: Real Time Vehicle Tracking in Uninstrumented Indoor Environments.** In *Proc. of the 13th ACM SenSys*, pages 99–112, 2015. 4, 11, 54, 78
 - [17] IGOR BOGOSLAVSKYI, LUCIANO SPINELLO, WOLFRAM BURGARD, AND CYRILL STACHNISS. **Where to Park? Minimizing the Expected Time to Find a Parking Space.** In *Proc. of the 2015 IEEE ICRA*, pages 2147–2152. IEEE, 2015. 6

-
- [18] ULRICH SCHWESINGER, MATHIAS BÜRKI, JULIAN TIMPNER, STEPHAN ROTTMANN, LARS WOLF, LINA MARIA PAZ, HUGO GRIMMETT, INGMAR POSNER, PAUL NEWMAN, CHRISTIAN HÄNE, ET AL. **Automated Valet Parking and Charging for E-Mobility**. In *Proc. of IEEE IV*, pages 157–164, 2016. 6
- [19] PIETRO EDOARDO CARNELLI, JOY YEH, MAHESH SOORIYABANDARA, AND AFTAB KHAN. **ParkUs: A Novel Vehicle Parking Detection System**. In *Proc. of the 31st AAAI*, pages 4650–4656, 2017. 7
- [20] MICHAEL JONES, AFTAB KHAN, PARAG KULKARNI, PIETRO CARNELLI, AND MAHESH SOORIYABANDARA. **ParkUs 2.0: Automated Cruise Detection for Parking Availability Inference**. In *Proc. of the 14th EAI Mobiquitous*, pages 242–251, 2017. 7
- [21] ROSARIO SALPIETRO, LUCA BEDOGNI, MARCO DI FELICE, AND LUCIANO BONONI. **Park Here! A Smart Parking System based on Smartphones’ Embedded Sensors and Short Range Communication Technologies**. In *Proc. of the 2nd IEEE WF-IoT*, pages 18–23, 2015. 7
- [22] MIKKEL BAUN KJÆRGAARD, MARTIN WIRZ, DANIEL ROGGEN, AND GERHARD TRÖSTER. **Detecting Pedestrian Flocks by Fusion of Multi-Modal Sensors in Mobile Phones**. In *Proc. of 14th ACM UbiComp*, pages 240–249, 2012. 7
- [23] YUKI MAEKAWA, AKIRA UCHIYAMA, HIROZUMI YAMAGUCHI, AND TERUO HIGASHINO. **Car-level Congestion and Position Estimation for Railway Trips Using Mobile Phones**. In *Proc. of the 16th ACM UbiComp*, pages 939–950, 2014. 7
- [24] NEHAL KASSEM, AHMED E KOSBA, AND MOUSTAFA YOUSSEF. **RF-based Vehicle Detection and Speed Estimation**. In *Proc. of the 75th IEEE VTC*, pages 1–5, 2012. 7
- [25] ANDROID AUTHORITY IAN SHERWIN. **Google Labs’ Open Spot: A Useful Application That No One Uses** [online]. 2011. <https://web.archive.org/web/20190119094651/https://www.androidauthority.com/google-labs-open-spot-a-useful-application-that-no-one-uses-15186>.

-
- [26] PRASHANTH MOHAN, VENKATA N PADMANABHAN, AND RAMACHANDRAN RAMJEE. **Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smartphones**. In *Proc. of the 6th ACM SenSys*, pages 323–336. ACM, 2008. 8, 9, 59
- [27] SASANK REDDY, MIN MUN, JEFF BURKE, DEBORAH ESTRIN, MARK HANSEN, AND MANI SRIVASTAVA. **Using Mobile Phones to Determine Transportation Modes**. *ACM Trans. on Sensor Netw.*, 6(2):13, 2010. 8, 9, 16
- [28] JEAN-GABRIEL KRIEG, GENTIAN JAKLLARI, HADRIEN TOMA, AND ANDRÉ-LUC BEYLOT. **Unlocking the Smartphone’s Sensors for Smart City Parking**. *Pervasive and Mobile Computing*, 43:78–95, 2018. 8
- [29] ANSHUL RAI, KRISHNA KANT CHINTALAPUDI, VENKATA N PADMANABHAN, AND RIJUREKHA SEN. **Zee: Zero-effort Crowdsourcing for Indoor Localization**. In *Proc. of the 18th ACM MobiCom*, pages 293–304. ACM, 2012. 8, 57
- [30] JIN WANG, NICHOLAS TAN, JUN LUO, AND SINNO JIALIN PAN. **WOLoc: WiFi-only Outdoor Localization Using Crowdsensed Hotspot Labels**. In *Proc. of the 36th IEEE INFOCOM*, 2017. 8, 57, 89
- [31] YI WANG, JIALIU LIN, MURALI ANNAVARAM, QUINN A JACOBSON, JASON HONG, BHASKAR KRISHNAMACHARI, AND NORMAN SADEH. **A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition**. In *Proc. of the 7th ACM MobiSys*, pages 179–192, 2009. 9, 16
- [32] JIE YANG, SIMON SIDHOM, GAYATHRI CHANDRASEKARAN, TAM VU, HONGBO LIU, NICOLAE CECAN, YINGYING CHEN, MARCO GRUTESER, AND RICHARD P. MARTIN. **Detecting Driver Phone Use Leveraging Car Speakers**. In *Proc. of the 17th ACM MobiCom*, pages 97–108, 2011. 9
- [33] YAN WANG, JIE YANG, HONGBO LIU, YINGYING CHEN, MARCO GRUTESER, AND RICHARD P MARTIN. **Sensing Vehicle Dynamics for Determining Driver Phone Use**. In *Proc. of the 11th ACM MobiSys*, pages 41–54. ACM, 2013. 9, 16, 59

-
- [34] ANDREAS BLOCH, ROBERT ERDIN, SONJA MEYER, THOMAS KELLER, AND ALEXANDRE DE SPINDLER. **Battery-Efficient Transportation Mode Detection on Mobile Devices**. In *Proc. of the 15th IEEE MDM*, pages 185–190, 2015. 9
- [35] JIANGCHUAN ZHENG, SIYUAN LIU, AND LIONEL M NI. **Effective Mobile Context Pattern Discovery via Adapted Hierarchical Dirichlet Processes**. In *Proc. of the 15th IEEE MDM*, pages 146–155, 2014. 9
- [36] STEVEN L. SCOTT. **Bayesian Methods for Hidden Markov Models: Recursive Computing in the 21st Century**. *Journal of the American Statistical Association*, **97**(457):337–351, 2002. 9, 33
- [37] SINNO JIALIN PAN AND QIANG YANG. **A Survey on Transfer Learning**. *IEEE Trans. on Knowledge and Data Eng.*, **22**(10):1345–1359, 2010. 9, 17, 25
- [38] DAVID PARDOE AND PETER STONE. **Boosting for Regression Transfer**. In *Proc. of the 27th ICML*, pages 863–870, 2010. 9, 25, 26
- [39] HARRIS DRUCKER. **Improving Regressors Using Boosting Techniques**. In *Proc. of the 14th ICML*, **97**, pages 107–115, 1997. 9, 25
- [40] YOAV FREUND AND ROBERT E SCHAPIRE. **A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting**. In *European conf. on comput. learn. theory*, pages 23–37. Springer, 1995. 9, 25
- [41] PARAMVIR BAHL AND VENKATA N PADMANABHAN. **RADAR: An In-Building RF-Based User Location and Tracking System**. In *Proc. of the 19th IEEE INFOCOM*, **2**, pages 775–784. IEEE, 2000. 10
- [42] YIFEI JIANG, XIN PAN, KUN LI, QIN LV, ROBERT P DICK, MICHAEL HANNIGAN, AND LI SHANG. **Ariel: Automatic Wi-Fi based Room Fingerprinting for Indoor Localization**. In *Proc. of the 2012 ACM Conf. on Ubiquitous Computing*, pages 441–450, 2012. 10

-
- [43] HE WANG, SOUVIK SEN, AHMED ELGOHARY, MOUSTAFA FARID, MOUSTAFA YOUSSEF, AND ROMIT ROY CHOUDHURY. **No Need to War-drive: Unsupervised Indoor Localization**. In *Proc. of the 10th ACM MobiSys*, pages 197–210, 2012. 10, 57
- [44] HEBA ABDELNASSER, REHAM MOHAMED, AHMED ELGOHARY, MOUSTAFA FARID ALZANTOT, HE WANG, SOUVIK SEN, ROMIT ROY CHOUDHURY, AND MOUSTAFA YOUSSEF. **SemanticSLAM: Using Environment Landmarks for Unsupervised Indoor Localization**. *IEEE Trans. on Mobile Computing*, **15**(7):1770–1782, 2016. 10, 57
- [45] MONTEMERLO MICHAEL, THRUN SEBASTIAN, K DAPHNE, AND W BEN. **FastSLAM 2.0: An Improved Particle Filtering Algorithm For Simultaneous Localization And Mapping That Provably Converges**. In *Proc. of the 16th IJCAI*, 2003. 10
- [46] RUIPENG GAO, MINGMIN ZHAO, TAO YE, FAN YE, YIZHOU WANG, KAIGUI BIAN, TAO WANG, AND XIAOMING LI. **Jigsaw: Indoor Floor Plan Reconstruction via Mobile Crowdsensing**. In *Proc. of the 2014 ACM MobiCom*, pages 249–260, 2014. 10
- [47] SARFRAZ NAWAZ AND CECILIA MASCOLO. **Mining Users’ Significant Driving Routes with Low-Power Sensors**. In *Proc. of the 12th ACM SenSys*, pages 236–250, 2014. 10, 59
- [48] SEBASTIAN THRUN, WOLFRAM BURGARD, AND DIETER FOX. *Probabilistic Robotics*. MIT press, 2005. 11
- [49] RAINER KÜMMERLE, RUDOLPH TRIEBEL, PATRICK PFAFF, AND WOLFRAM BURGARD. **Monte Carlo Localization in Outdoor Terrains using Multilevel Surface Maps**. *Journal of Field Robotics*, **25**(6-7):346–359, 2008. 11
- [50] STEFFEN WAHL, PETER SCHLUMBERGER, RAUL ROJAS, AND MARTIN STAMPFLE. **Localization Inside a Populated Parking Garage by Using Particle Filters With a Map of the Static Environment**. In *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pages 95–100, 2015. 11

-
- [51] FELIX WERNER, FREDERIC MAIRE, JOAQUIN SITTE, HOWIE CHOSSET, STEPHEN TULLY, AND GEORGE KANTOR. **Topological SLAM Using Neighbourhood Information of Places**. In *Proc. of the 2009 IEEE IROS*, pages 4937–4942, 2009. 11
- [52] FENG LU AND EVANGELOS MILIOS. **Globally Consistent Range Scan Alignment for Environment Mapping**. *Autonomous Robots*, 4(4):333–349, 1997. 11
- [53] GIORGIO GRISETTI, RAINER KUMMERLE, CYRILL STACHNISS, AND WOLFRAM BURGARD. **A Tutorial on Graph-Based SLAM**. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010. 11, 54, 68
- [54] RAINER KUMMERLE, DIRK HAHNEL, DMITRI DOLGOV, SEBASTIAN THRUN, AND WOLFRAM BURGARD. **Autonomous Driving in a Multi-level Parking Structure**. In *Proc. of the 2009 IEEE ICRA*, pages 3395–3400, 2009. 11, 53
- [55] SEBASTIAN THRUN AND MICHAEL MONTEMERLO. **The Graph SLAM Algorithm with Applications to Large-scale Mapping of Urban Structures**. *Int. Journal of Robotics Research*, 25(5-6):403–429, 2006. 11, 68, 84
- [56] JOSEPH HUANG, DAVID MILLMAN, MORGAN QUIGLEY, DAVID STAVENS, SEBASTIAN THRUN, AND ALOK AGGARWAL. **Efficient, Generalized Indoor WiFi GraphSLAM**. In *Proc. of the 2011 IEEE ICRA*, pages 1038–1043, 2011. 11
- [57] PIOTR MIROWSKI, TIN KAM HO, SAEHOON YI, AND MICHAEL MACDONALD. **SignalSLAM: Simultaneous Localization and Mapping with Mixed WiFi, Bluetooth, LTE and Magnetic Signals**. In *Proc. of the 2013 IEEE IPIN*, pages 1–10, 2013. 11
- [58] SAEHOON YI, PIOTR MIROWSKI, TIN KAM HO, AND VLADIMIR PAVLOVIC. **Pose Invariant Activity Classification for Multi-floor Indoor Localization**. In *Proc. of the 2014 IEEE ICPR*, pages 3505–3510, 2014. 11
- [59] YURONG JIANG, HANG QIU, MATTHEW MCCARTNEY, GAURAV SUKHATME, MARCO GRUTESER, FAN BAI, DONALD GRIMM, AND RAMESH GOVINDAN.

-
- CARLOC: Precise Positioning of Automobiles.** In *Proc. of the 13th ACM SenSys*, pages 253–265, 2015. 11, 54, 57
- [60] JULIAN R ULLMANN. **An Algorithm for Subgraph Isomorphism.** *Journal of the ACM (JACM)*, **23**(1):31–42, 1976. 12
- [61] BRENDAN D MCKAY ET AL. **Practical Graph Isomorphism.** *Congressus Numerantium*, **30**:45–87, 1981. 12
- [62] LUIGI P CORDELLA, PASQUALE FOGGIA, CARLO SANSONE, AND MARIO VENTO. **A (Sub) Graph Isomorphism Algorithm for Matching Large Graphs.** *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **26**(10):1367–1372, 2004. 12, 54, 64, 66
- [63] RICHARD ARNOTT AND EREN INCI. **An Integrated Model of Downtown Parking and Traffic Congestion.** *Elsevier Journal of Urban Economics*, **60**(3):418–442, 2006. 14
- [64] H. ALY, A. BASALAMAH, AND M. YOUSSEF. **LaneQuest: An Accurate and Energy-Efficient Lane Detection System.** In *Proc. of the 13th IEEE PerCom*, pages 163–171, 2015. 16
- [65] BÜLENT ÜSTÜN, WILLEM J MELSSSEN, AND LUTGARDE MC BUYDENS. **Facilitating The Application of Support Vector Regression by Using A Universal Pearson VII Function Based Kernel.** *Chemometrics and Intelligent Laboratory Systems*, **81**(1):29–40, 2006. 21, 24
- [66] ALEX J SMOLA AND BERNHARD SCHÖLKOPF. **Tutorial on Support Vector Regression.** *Statistics and Computing*, **14**(3):199–222, 2004. 23
- [67] CHUN-HSIN WU, JAN-MING HO, AND DER-TSAI LEE. **Travel-Time Prediction with Support Vector Regression.** *IEEE Trans. on Intelligent Transportation Systems*, **5**(4):276–281, 2004. 23
- [68] KARL PEARSON. **Contributions to The Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material.** *Philosophical Trans. of the Royal Society of London. A*, pages 343–414, 1895. 24

-
- [69] KARTIK MURALIDHARAN, AZEEM JAVED KHAN, ARCHAN MISRA, RAJESH KRISHNA BALAN, AND SHARAD AGARWAL. **Barometric Phone Sensors: More Hype Than Hope!** In *Proc. of the 15th ACM HotMobile*, page 12, 2014. 29
- [70] LEO BREIMAN. **Random Forests**. *Machine Learning*, **45**(1):5–32, 2001. 31
- [71] F. JELINEK, L. BAHL, AND R. MERCER. **Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech**. *IEEE Trans. on Information Theory*, **21**(3):250–256, 1975. 33
- [72] RAJJAN SHINGHAL AND GODFRIED T. TOUSSAINT. **Experiments in Text Recognition with the Modified Viterbi Algorithm**. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **1**(2):184–193, 1979. 33
- [73] MARK A HALL AND LLOYD A SMITH. **Practical Feature Subset Selection for Machine Learning**. In *Proc. of 21th ACSC*, pages 181–191, 1998. 36, 46
- [74] GUANG-BIN HUANG, QIN-YU ZHU, AND CHEE-KHEONG SIEW. **Extreme Learning Machine: Theory and Applications**. *Neurocomputing*, **70**(1-3):489–501, 2006. 48
- [75] BERNARDINO ROMERA-PAREDES AND PHILIP TORR. **An Embarrassingly Simple Approach to Zero-Shot Learning**. In *Proc. of the 32nd ICML*, pages 2152–2161, 2015. 49
- [76] YU-XIONG WANG AND MARTIAL HEBERT. **Learning to Learn: Model Regression Networks for Easy Small Sample Learning**. In *Proc. of the 14th ECCV*, pages 616–634, 2016. 49
- [77] ELENI TRIANTAFILLOU, RICHARD ZEMEL, AND RAQUEL URTASUN. **Few-Shot Learning Through an Information Retrieval Lens**. In *Proc. of the 31st NIPS*, pages 2255–2265, 2017. 49
- [78] INSURANCE.COM. **Biggest Driving Embarrassments: Forgetting where one Parked and Driving over Curbs** [online]. 2014. <https://web.archive.org/web/20170920161822/http://www.insurance.com/press-room/biggest-driving-embarrassment>. 53

-
- [79] ROY WANT. **Near Field Communication.** *IEEE Pervasive Computing*, **10**(3):4–7, 2011. 53
- [80] KIRTI CHAWLA, CHRISTOPHER MCFARLAND, GABRIEL ROBINS, AND CONNOR SHOPE. **Real-time RFID Localization Using RSS.** In *Proc. of the 2013 IEEE Int. Conf. on Localization and GNSS*, pages 1–6. IEEE, 2013. 53
- [81] ZHUOLING XIAO, HONGKAI WEN, ANDREW MARKHAM, AND NIKI TRIGONI. **Lightweight Map Matching for Indoor Localisation Using Conditional Random Fields.** In *Proc. of the 13th IEEE IPSN*, pages 131–142, 2014. 53
- [82] YUNYE JIN, HONG-SONG TOH, WEE-SENG SOH, AND WAI-CHOONG WONG. **A Robust Dead-Reckoning Pedestrian Tracking System with Low Cost Sensors.** In *Proc. of the 2011 IEEE PerCom*, pages 222–230, 2011. 53
- [83] RA TELL, G SIAS, J SMITH, J SAHL, AND R KAVET. **ELF Magnetic Fields in Electric and Gasoline-powered Vehicles.** *Bioelectromagnetics*, **34**(2):156–161, 2013. 53
- [84] MICHAEL J CARUSO AND LUCKY S WITHANAWASAM. **Vehicle Detection and Compass Applications Using AMR Magnetic Sensors.** In *Sensors Expo Proceedings*, **477**, page 39, 1999. 53
- [85] JINSOO LEE, WOOK-SHIN HAN, ROMANS KASPEROVICS, AND JEONG-HOON LEE. **An In-depth Comparison of Subgraph Isomorphism Algorithms in Graph Databases.** *Proc. VLDB Endow.*, **6**(2):133–144, December 2012. 54
- [86] YU-CHUNG CHENG, YATIN CHAWATHE, ANTHONY LAMARCA, AND JOHN KRUMM. **Accuracy Characterization for Metropolitan-scale Wi-Fi Localization.** In *Proc. of the 3rd ACM MobiSys*, pages 233–245. ACM, 2005. 57
- [87] JIM CHERIAN, JUN LUO, HONGLIANG GUO, SHEN-SHYANG HO, AND RICHARD WISBRUN. **ParkGauge: Gauging the Occupancy of Parking Garages with Crowdsensed Parking Characteristics.** In *Proc. of the 17th IEEE MDM*, pages 92–101, 2016. 58
- [88] THOMAS D GILLESPIE. *Fundamentals of Vehicle Dynamics.* Society of Automotive Engineers (SAE) International, 1997. 59

-
- [89] HAOFU HAN, JIADI YU, HONGZI ZHU, YINGYING CHEN, JIE YANG, YANMIN ZHU, GUANGTAO XUE, AND MINGLU LI. **SenSpeed: Sensing Driving Conditions to Estimate Vehicle Speed In Urban Environments**. In *Proc. of the 33rd IEEE INFOCOM*, pages 727–735, 2014. 59
- [90] J. ROSS QUINLAN. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. 59
- [91] CHI ZHANG, KALYAN P SUBBU, JUN LUO, AND JIANXIN WU. **GROPING: Geomagnetism and Crowdsensing Powered Indoor Navigation**. *IEEE Trans. on Mobile Computing*, **14**(2):387–400, 2015. 59
- [92] ALAN SAALFELD. **Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm**. *Cartography and Geographic Information Science*, **26**(1):7–18, 1999. 62
- [93] QIAN ZHOU, FAN YE, XIAOGE WANG, AND YUANYUAN YANG. **Automatic Construction of Garage Maps for Future Vehicle Navigation Service**. In *Proc. of the 2016 IEEE ICC*, pages 1–7, 2016. 83

Publications

JOURNAL ARTICLES

- 1) **Jim Cherian**, Jun Luo, and Shen-Shyang Ho. ParkLoc: Light-weight Graph-based Vehicular Localization in Parking Garages. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2.3 (2018): 99.
- 2) **Jim Cherian**, Jun Luo, Hongliang Guo, Shen-Shyang Ho, and Richard Wisbrun. ParkGauge+: Towards Scalable Occupancy Inference for Parking Garages. *IEEE Transactions on Mobile Computing*. (*Under review*, Initial submission: May 2017, Major revision: Dec 2018)

CONFERENCE PAPERS

- 3) **Jim Cherian**, Jun Luo, and Shen-Shyang Ho. ParkLoc: Light-weight Graph-based Vehicular Localization in Parking Garages. 2018 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'18), Singapore, Oct 2018.
- 4) **Jim Cherian**, Jun Luo, Hongliang Guo, Shen-Shyang Ho, and Richard Wisbrun. ParkGauge: Gauging the Occupancy of Parking Garages with Crowdsensed Parking Characteristics. In *Proceedings of the 17th IEEE Conference on Mobile Data Management (MDM'16)*, pp. 92-101, Porto, Portugal, June 2016.
- 5) **Jim Cherian**, Jun Luo, Hongliang Guo, Shen-Shyang Ho, and Richard Wisbrun. Poster: ParkGauge: Gauging the Congestion Level of Parking Garages with Crowdsensed Parking Characteristics. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys'15)*, pp. 395-396,

Seoul, South Korea, November 2015.

- 6) Wye Huong Yan, Justin Ong, Shen-Shyang Ho, and **Jim Cherian**. Traffic incident validation and correlation using text alerts and images. In Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'14), pp. 601-604, Dallas, Texas, USA, November 2014.

PATENTS

- 7) **Jim Cherian**, Jun Luo, Hongliang Guo, Shen-Shyang Ho, and Richard Wisbrun. Verfahren zur Ermittlung eines Belegungsgrades einer Parkeinrichtung (Method to determine the occupancy rate of a parking garage. Germany Patent DE102015220907. October 2015.