ABSTRACT

Title of dissertation: UBIQUITOUS ACCESSIBILITY DIGITAL-MAPS FOR SMART CITIES: PRINCIPLES AND REALIZATION

Heba A. Ismail
Doctor of Philosophy, 2019

Dissertation directed by: Professor Ashok Agrawala
Department of Computer Science

To support disabled individuals' active participation in the society, the Americans with Disabilities Act (ADA) requires installing various accessibility measures in roads and public accommodation spaces such as malls and airports. For example, curb ramps are installed on sidewalks to aid wheel-chaired individuals to transition from/to sidewalks smoothly. However, to comply with the ADA requirements, it is sufficient to have one accessible route in a place and usually there are no clear directions on how to reach that route. Hence, even within ADA-compliant facilities, accessing them can still be challenging for a disabled individual. To improve the spaces' accessibility, recently, systems have been proposed to rate outdoor walkways and intersections' accessibility through active crowdsourcing where individuals mark and/or validate a maps' accessibility assessments. Yet, depending on humans limits the ubiquity, accuracy and the update-rate of the generated maps.

In this dissertation, we propose the AccessMap—Accessibility Digital Maps—system to build ubiquitous accessibility digital-maps automatically; where indoor/outdoor

spaces are updated with various accessibility semantics and marked with assessment of their accessibility levels for the vision- and mobility-impairment disability types. To build the maps automatically, we propose a passive crowdsourcing approach where the users' smartphone devices' spatiotemporal sensors signals (e.g. barometer, accelerometer, etc.) are analyzed to detect and map the accessibility semantics. We present algorithms to passively detect various semantics such as accessible pedestrian signals and missing curb-ramps. We also present a probabilistic framework to construct the map while taking the uncertainty in the detected semantics and the sensors into account. AccessMap was evaluated in two different countries, the evaluation results show high detection accuracy for the different accessibility semantics. Moreover, the crowdsourcing framework helps further improve the map integrity overtime.

Additionally, to tag the crowdsourced data with location stamps, GPS is the de-facto-standard localization method, but it fails in indoor environments. Thus, we present the Hapi WiFi-based localization system to estimate the crowdsourcers' location indoors. WiFi represents a promising technology for indoor localization due to its world-wide deployment. Nevertheless, current systems either rely on a tedious expensive offline calibration phase, are based on heuristics-based/theoretical approaches that limit their accuracy, and/or focus on a single-floor area of interest. To address these limitations, Hapi combines signal-processing, deep-learning and probabilistic models to estimate a user's 2.5D location (i.e. the user floor-level and her 2D location within that floor) in a calibration-free manner. Our evaluation results show that, in high-rise buildings, we could achieve significant improvements

over state-of-the-art indoor-localization systems.

UBIQUITOUS ACCESSIBILITY DIGITAL-MAPS FOR SMART CITIES: PRINCIPLES AND REALIZATION

by

Heba A. Ismail

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Professor Ashok Agrawala, Chair/Advisor
Professor Moustafa Youssef
Professor A. Udaya Shankar
Professor Adam Porter
Professor Min Wu (Dean's Representative)

Dedication

To all girls who want to make a difference

and pursue a path they choose.

Acknowledgments

I am indebted to my advisors Prof. Ashok Agrawala and Prof. Moustafa Youssef for their constant support, invaluable guidance and encouragement. I have learned a lot from them—not just professionally but also personally. Going through a car accident, Prof. Agrawala made sure to take care of my mental- and physical-health, he taught me how to be a caring supervisor while making sure that I expand my vision and look at the big picture. Prof. Moustafa's hard work, and dedication to research has been truly inspirational since I was an undergraduate student. He taught me how to become a researcher and encouraged me to have a voice.

I would like to thank Prof. Atif Memon whose thoughtfulness and advice meant so much to me. Prof. Atif is a source of encouragement and support, not just for me, but for lots of students in the department. I am also grateful to professors A. Udaya Shankar, Adam Porter and Min Wu for being on my dissertation committee. Special thanks are due to Dr. John Krumm for his guidance and support. It has been a pleasure to work with and learn from such an extraordinary individual. I would like to also thank Prof. Gireeja Ranade and Dr. Eric Horvitz for their advice during and after my internship in Microsoft Research.

I would like to thank my friends in Maryland for making me feel part of a group and filling my PhD years with great memories that I will always cherish and smile when I look back at them. In particular, I am really grateful to Shravan Srinivasan for his care, support, and encouragement. Warm thanks are due to Nishant Rodrigues for being a great source of support and care. I don't think I can

express my gratitude in words for Shravan and Nishant. I owe a big thanks to Nirat Saini for being there whenever I needed her, and being a great friend to me. I would like to also thank Ananth Hari for his care and his help with my talks' preparation; and Bhargavi Patel for her support. Though, I got to become friends with Bhargavi shortly before she left, she has always shown me lots of care, understanding, and support. Warmest thanks are due to Mahfuza Sharmin for her unwavering support from my first day in Maryland till I finished my PhD program.

During the course of my PhD study, I have got to know lots of fellow students, flatmates, and labmates. I would like to acknowledge their role in enriching my experiences. I have learned a lot from them, and they helped me mature and grow.

Last but not least, I am very grateful to my parents, my sister, Hend, and my uncle, Hany, for their constant love, understanding, and making me smile while remotely sharing my PhD journey. I am deeply indebted to Hend for her support, and for the love and warmness she and her baby gave me.

Table of Contents

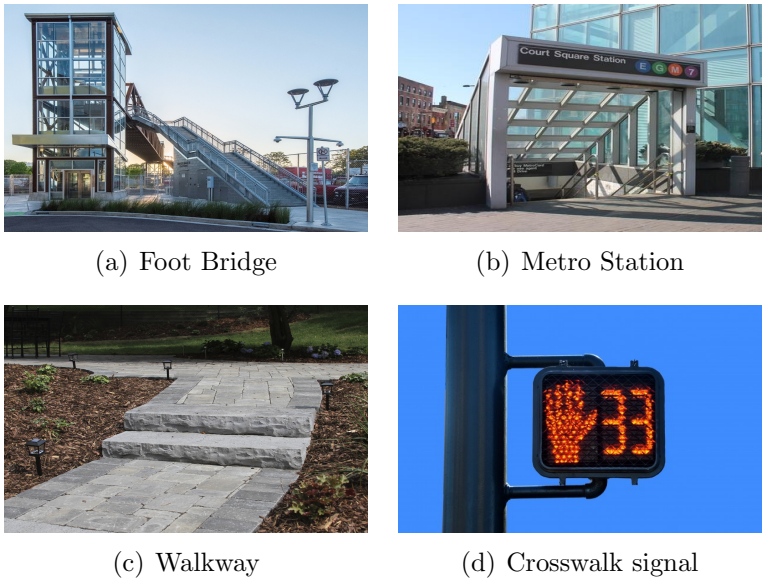
List of Tables

List of Figures

Chapter 1: Introduction

1.1 Accessibility Maps

One in every five Americans reported having a disability according to the Census Bureau Reports [1]. Individuals can experience different types of disabilities, temporally or permanently, including physical, sensory or a combination of them. These disabilities affect how they can independently commute comfortably; as different spaces' structure and/or functionality can affect their ability to access them (Figure 1.1).



(a) Foot Bridge (b) Metro Station

(c) Walkway (d) Crosswalk signal

Figure 1.1: Various semantics can affect a space's accessibility for the different disability types.

Thus, to ensure that disabled individuals have equal access to the different spaces, the Americans with Disabilities Act (ADA) sets minimum requirements for roads and buildings accessibility designs [2]. For example, ramps should be deployed to provide a smooth transition between elevation changes on a sidewalk, instead or along with a staircase, and to allow wheel-chaired individuals to safely use the sidewalk as shown in Figure 1.2 below. Another example is deploying accessible pedestrian signals which communicate information about the "Walk" and "Don't walk" intervals at signalized intersections in non-visual formats to help pedestrians who are blind or who have low vision commute independently.

(a) Wheel-chaired **inaccessible** (Staircase on a sidewalk).

(b) Wheel-chaired **accessible** (Ramp on a sidewalk).

Figure 1.2: Example for an inaccessible route vs. an accessible one for wheel-chaired individuals.

Deploying such accessibility measures in smart-cities leads to improvements in their citizens' human rights and impactful outcomes across key services such as healthcare, transportation, and education. However, to utilize the accessibility measures and improve disabled individuals' quality of life, mapping the measures is imperative. For instance, using the resultant accessibility map, a disabled individual

can find accessible routes for her disability type(s); choose suitable/accessible areas to move-in/visit; among others [3]. Nevertheless, while digital-maps have evolved in recent-years to become a part of our day-to-day life, yet, both commercial and open-source maps still miss lots of road semantics [4, 5] and have limited accessibility information—if any. As these maps have been traditionally constructed through satellite images, road surveyors, and/or manual entry by trained personnel; it is hard to keep them up-to-date and capture the dynamic and rich accessibility information. Thus, a crowd-sourcing solution is inevitable to achieve ubiquitous coverage efficiently [5–7].

Recently, there has been some recent efforts to build accessibility maps using crowdsourcing [7–11]. However, these systems relied on active crowdsourcing where volunteers or paid Amazon Mechanical Turk workers (turkers) manually and/or semi-manually identify sidewalk accessibility problems in Google Street View imagery [12]. While the approaches help construct accessibility maps, the constructed maps are limited in spatial and disability coverage. A truly ubiquitous accessibility map construction system should achieve the following:

- Indoor and outdoor coverage: The accessibility map needs to rate the accessibility level for both indoor and outdoor spaces.
- Automatic construction: For the accessibility map to realistically have a ubiquitous coverage, it needs to remove any manual intervention in the data collection and/or accessibility assessment.
- High accuracy: Disabled individuals are vulnerable population, inaccuracies

in the map can lead not to just inconvenience to the user, but incidents also.

- Support different disabilities: There is a wide-range of disabilities, the map construction system should support and easily enable supporting different disabilities.

In this dissertation, we describe a passive crowdsourcing-based system, *AccessMap*, to build the accessibility maps ubiquitously automatically. We also describe *AccessMap*'s enabling components including how to obtain the crowd-sourcers location indoors and how to update the map to improve its accuracy and keep it up-to-date.

1.2 Dissertation Research Approach and Overview

The *AccessMap* system automatically builds ubiquitous accessibility digital-maps; where indoor and outdoor spaces are automatically updated with various accessibility semantics and marked with assessment of their accessibility state for the vision-impairment and mobility disability types. We define an accessibility semantic as either:

1. A measure specially deployed in a space to improve the space's accessibility for one or more disability types. For example, curb-ramps deployed at street intersections which allow wheelchair users to move onto or off a sidewalk without difficulty.

2. A part of a space that makes the space inaccessible for one or more disability

types. For example, a staircase on a sidewalk which hinders wheel-chaired individuals from using that segment of the sidewalk.

Nowadays mobile devices come with a growing list of embedded sensors. Our analysis shows that these sensors get affected by the user's environment and how she interacts with it. For example, the acceleromter sensor gets affected by the acceleration force excreted while walking which differs when walking on a flat surface as opposed to ascending/descending a staircase as shown in Figure 1.3. The figure shows the x acceleration from the user mobile device while walking on a flat surface till the 8th second then the user descends a staircase leading to higher acceleration magnitude.

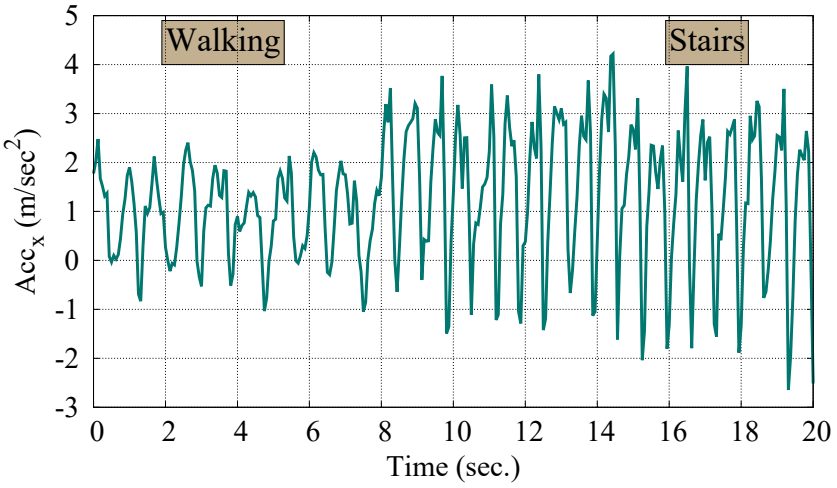


Figure 1.3: Example showing the effect of descending stairs on a sidewalk on the acceleration signal.

Therefore, *AccessMap* can leverage accelerometers to detect staircases on sidewalks. Once, a staircase is detected, it can, then, assess this part of the sidewalk as inaccessible for wheel-chaired individuals as we can see in Figure 1.4 below.

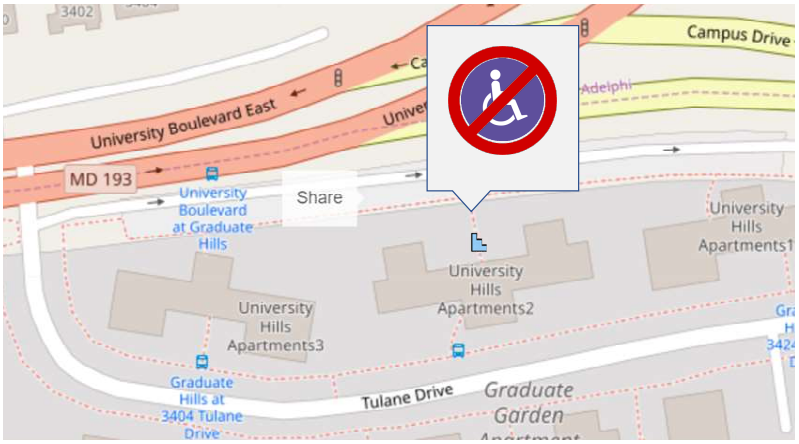


Figure 1.4: Example showing mapping a staircase to a sidewalk and assessing the sidewalk as wheel-chair inaccessible.

AccessMap builds on this analysis and crowd-sources time and location stamped multi-model sensors (E.g. accelerometer and microphone) from the users' smart-phones to discover a wide-range of accessibility semantics. Subsequently, *AccessMap* leverages the detected semantics to rate the accessibility-state of the different indoor and outdoor spaces for a given disability type.

Furthermore, to construct the map and keep it up-to-date, *AccessMap* employs a probabilistic framework to update the digital maps with the detected accessibility semantics. The framework handles the heterogeneity and uncertainty in the detected measures and the crowd-sensed location data which helps rapidly refine the map accuracy. Figure 1.5, below, shows the *AccessMap* system architecture overview.

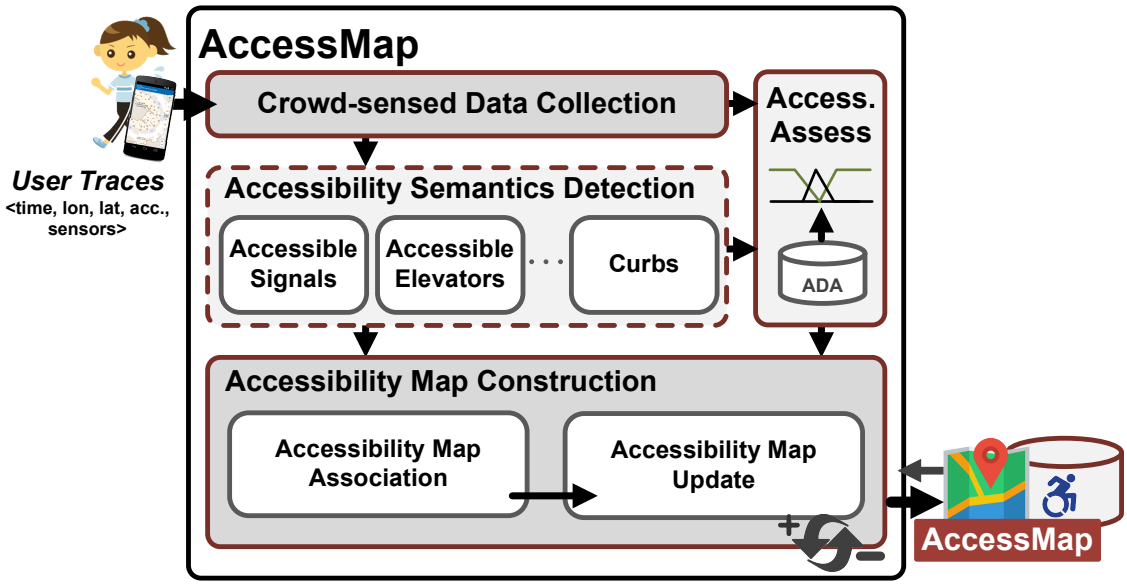


Figure 1.5: The *AccessMap* system architecture overview.

AccessMap is based on a passive crowdsensing approach, where the *Crowdsensed Data Collection* Module collects various sensor measurements from the users' smartphones, and submit it to the *AccessMap* service running in the cloud in a way transparent to the users. Then, the *Accessibility Semantics Detection* Module analyzes the crowdsensed data to identify the availability or lack of accessibility measures for the different disability types. The detected semantics are employed to assess the spaces' accessibility state as **accessible** or **inaccessible** via the *Accessibility Assessment* Module. Finally, the *Accessibility Map Update* Module constructs the accessibility map.

1.2.1 CrowdSensed Data

The *AccessMap* system collects time- and location-stamped traces along with sensor measurements including accelerometer and barometer. These sensors have a

low cost energy profile and they are already running all the time during the standard phone operation. Thus, consuming them adds zero extra energy. In addition, based on the user location and mobility context, the phone's microphone is turned-on to record ambient sounds.

For the location stamp, while GPS is widely available on all smartphone devices, GPS fails indoors due to its satellites signal attenuation limiting the number of satellites in-view. Thus, relying on other technologies is inevitable. WiFi represents an attractive technology due to WiFi infrastructure world-wide ubiquitous deployment in almost all buildings and WiFi chips availability on all smartphone devices for internet connectivity. Thus, to estimate a crowdsourcer location indoors, we rely on her smartphone device WiFi information. The *Hapi* indoor location estimation system is presented in Chapter 3. The *Hapi* system estimates the users' 2.5D location (her building floor number and 2D location within that floor) using her smartphone's visible WiFi accesspoints and their received signal strengths.

1.2.2 Accessibility Assessment

The *Accessibility Semantics Detection* Module is *AccessMap*'s core module, it is responsible for passively detecting the different accessibility semantics. Specifically, the module applies machine-learning techniques, such as classical supervised classification techniques and deep-learning neural networks, to accurately detect the following *accessibility measures*: ramps, accessible pedestrian signals, elevators, and accessible elevators. Additionally, it detects the following *accessibility issues*:

staircases and curbs. Chapter 5 details the detection algorithm for the accessibility semantics. Based on the detected semantics, *AccessMap* assesses the space's accessibility for wheel-chaired and visually-impaired individuals. We also discuss the assessment in Chapter 5.

1.2.3 Accessibility Map Construction

The *Accessibility Map Construction* module is responsible for constructing the accessibility map. *AccessMap* employs a probabilistic framework to build and refine the map incrementally as we get more crowdsourced data. The map construction and update algorithm is presented in Chapter 6.

1.3 Summary of Contributions

The contributions of this dissertation are summarized as follows:

- We present the *AccessMap* system architecture to automatically build a ubiquitous accessibility map where the various semantics are mapped and the spaces' accessibility-state are assessed for the different disability types.
- We provide methods for passively sensing various accessibility semantics such as accessible traffic signals, accessible elevators and missing curb-ramps.
- We provide a probabilistic framework to build the accessibility maps incrementally from the crowdsourced data.
- We show an implementation of *AccessMap* on Android devices and evaluate

its accuracy and energy-efficiency in multiple countries.

- We present *Hapi* as a calibration-free WiFi-based indoor localization system that takes into account visible APs from all floors to estimate the user floor-level and her 2D location on that floor.

- We present a deep-learning-based floor-level estimation method which only requires the building's WiFi APs location for training and deployment. We believe the presented method is independent from *Hapi* and can be used to estimate the user floor level for other location-based services.

- We present an RSS-Rank probabilistic-based algorithm that estimates the user location using her current floor-level and visible APs from all floors.

- We present a regression-based method for predicting *AccessMap*'s user-location estimates' quality.

- We present a Kalman-Filter based method to further refine the *Hapi* location estimate accuracy.

- We show the implementation of the *Hapi* system on a wide-range of Android devices and thoroughly evaluate it in a diverse set of multi-story buildings with 13 subjects over 6 months.

- We give a side-by-side comparison of *Hapi* with two state-of-the-art calibration-free indoor localization systems.

We also discuss our insights and identify opportunities for future research.

1.4 Dissertation Roadmap

The rest of this dissertation is organized as follows: We start by discussing background for smartphone-based sensing, surveying accessibility maps construction systems, automatic semantics maps construction technologies and localization systems in Chapter 2. In Chapter 3, we present our smartphone-based indoor localization system. We show how to estimate the user's floor-level from her smartphone visible accesspoints information and how to estimate her 2D location on that floor using visible accesspoints from all floors. Chapter 4 shows *Hapi*'s implementation and evaluation. Thereafter, we present algorithms to passively detect various accessibility semantics in Chapter 5. In Chapter 6, we present a probabilistic algorithm to construct and update *AccessMap*'s accessibility map from the crowdsourced data. Chapter 7 gives *AccessMap* implementation and evaluation. Finally, Chapter 8 concludes the dissertation and discusses future work.

Chapter 2: Background and Related Work

2.1 Overview

In this chapter, we survey research work related to digital accessibility maps construction. We start by presenting systems that work on building accessibility maps. Then, we discuss algorithms for automatic digital map updates and automatic semantic maps enrichment. After that, we move to discussing smartphone-based outdoor and indoor localization systems.

2.2 Accessibility Maps Construction

The vulnerability of the disabled individuals and the lack of means to determine road accessibility apriori, triggered some recent efforts to build accessibility maps [7–11]. The IBM Sidewalks [8] presented a crowdsourcing system where citizens can report worn sidewalks or obstructed ones, that they come across, to city administrators via their phones. While IBM Sidewalks depended on volunteer citizens facing sidewalk problems, Hara et al. [7] investigated the feasibility of using untrained crowd-workers from Amazon Mechanical Turk (turkers) to manually identify sidewalk accessibility problems in Google Street View imagery [12]. In the study,

turkers were able to detect wheel-chaired accessibility for different spaces with high accuracy (around 80%) and it was further improved (around 90%) through an additional manual quality control stage. Yet, a clear problem with these approaches is the human time-cost of the manual labeling and quality control phases. Thus, Tohme [9] was proposed to automate part of the process through applying computer vision and machine-learning techniques. Specifically, a classical computer vision technique is applied to Google street view images to determine whether a curb-ramp is missing or not. Then, an SVM is used to predict the quality of the technique's result. Based on the prediction, the image is sent to the appropriate turker task (labeling or verification). While Tohme's pipeline achieved comparable accuracy, it has a small reduction in the human time-cost. This is due to the low precision of the employed computer-vision technique. To address this problem, in [10], authors proposed a Convolutional Neural Network (CNN) to find street regions where curb ramps are missing. Nevertheless, due to the difficulty of the problem and the data limitations, the CNN had around 30% recall only. Similar to Tohme, in [11], authors apply computer vision techniques on Google Street Views and Satellite Images to detect the zebra cross-walks and help visually-impaired individuals.

All these systems rely on Google Street Views to automate building the accessibility maps. As a result, they inherit a number of limitations:

- First, using the street views, one image is available for a space, which limits the accuracy of the system and makes it mandatory to turn to humans for manual labeling/verification for improving the accuracy.

- Second, Google Street Views are not available indoors which limits the coverage/ubiquity of the resultant maps.
- Third, Google Street Views can be quite outdated in certain areas which limits the accuracy/integrity of the generated maps.
- Finally, depending on images limits the maps to visual accessibility measures only.

On the other hand, in this dissertation, we present *AccessMap* which crowdsources multimodal sensors widely available on mobile devices to build the accessibility maps indoors and outdoors. The sensors' different modalities help identify various non-visual accessibility measures. Additionally, as *AccessMap* employs a passive crowdsensing approach, it truly automates the accessibility map construction.

Table 2.1 highlights the main differences between *AccessMap* and state-of-the-art accessibility map construction systems.

System	Disabilities-supported	Accessibility Semantics	Indoors	Outdoors	Automatic/Manual
IBM Sidewalks [8]	Wheel-chaired, visually-impaired	Obstructed and worn sidewalks	No	Yes	Manual
Thome [9]	Wheel-chaired	Curb-ramps	No	Yes	Semi-manual
Ahmetovic et al. [11]	Visually-impaired	Zebra crossing	No	Yes	Automatic
AccessMap	Wheel-chaired, visually-impaired	Curb-ramps, elevators, audio-enabled elevators, audio-enabled signals, curbs, stairs, ramps	Yes	Yes	Automatic

Table 2.1: Summary comparison between *AccessMap* and related accessibility map construction systems.

2.3 Digital Map and Automatic Semantics Updates

Automatic digital-map construction is a promising solution for capturing the dynamics of the real-world rich semantics. Recent work proposed to update the maps and even infer road semantics from GPS traces. For example, CrowdAtlas [6] uses map-matching to update OpenStreetMap's road network and add missing roads from GPS traces. Similarly, in [13] and [14] authors presented map-update systems to augment the road network's connectivity by adding intersections and road types respectively. In [15], authors proposed to use crowdsourced position and velocity traces to update maps with predicted traffic signals state. To further enrich the maps, inertial sensors were recently utilized to detect different road features. For instance, the Pothole Patrol [16] and Nericell [17] systems used accelerometers to detect potholes and the traffic condition. However, both systems used special accelerometer sensors which have higher sampling rates and lower noise as compared to typical chips available on phones. More recent systems used smartphone sensors to detect the road features, such as CrowdWatch [18] and Map++ [4, 19]. The CrowdWatch [18] system proposed to use accelerometer and orientation sensors to detect sidewalk temporary obstacles, e.g. temporary parking, via crowdsourcing and then remind distracted walkers about the dangers in front of them as they approach a dangerous area. On the other hand, the Map++ system [4, 19] is the most similar system to *AccessMap*; it utilizes accelerometer, magnetometer, orientation and cellular connectivity to identify a number of vehicular and pedestrian semantics. However, compared to Map++, *AccessMap* utilizes a larger set of sensors to

detect a more fine-grained semantics, i.e. accessibility measures installed at different places. Additionally, *AccessMap* provides a thoroughly analyzed automatic map-construction and update algorithm.

Another difference is that, while these systems focus on outdoor road maps, *AccessMap* covers both indoor and outdoor spaces. While there has been efforts to build indoor maps, e.g. [20] where authors proposed using sensors available on smartphones to build indoor floor-plans and add semantics like escalators, *AccessMap* extends such maps by adding the accessibility information. Moreover, *AccessMap* employs a probabilistic framework to handle the uncertainty and heterogeneity in the input data and provide a principled way to update the map as we get more crowd-sourced data.

2.4 Smartphone-based Outdoor Localization Systems

The Global Position System (GPS) [21] gives an accurate location information in outdoor environments. However, it suffers from large power consumption and accuracy degradation when the satellite signals are weak, e.g. due to tall buildings surrounding the road (i.e. inside urban canyons), inside tunnels, and bad weather conditions [22]. Even with sufficient satellites in-view, the geometric dilution of precision can often be large causing the estimated location points to be scattered around the original location [23], especially for the low-cost GPS chips typically available in low-end smartphones.

Alternative localization techniques have been introduced to address the GPS

issues including city-wide GSM and WiFi localization that trade energy-efficiency with less accurate localization. Typically these systems are cell-ID based [24] or depend on the RF-transmitters' received signal strength information [25–32] as both are available for the application developers, compared to e.g. angle-of-arrival based systems. Due to the complex signal propagation effects, RSS-based techniques require the construction of a fingerprint map with RSS signatures for locations inside the area of interest. This calibration step is an expensive overhead, especially when covering large areas. Also, due to the transmitters range and their deployment density [33, 34], such systems' localization accuracy can vary from hundreds of meters to kilometers.

Recently, inertial sensors have become main-stream in smartphone devices. Researchers have leveraged these low-energy sensors to achieve energy-efficient localization [35–38]. The main idea is to use the smartphone's accelerometer, magnetometer and gyroscope (if available) sensors to dead-reckoning the device location and use other sensors to reset the deadreckoning rapidly accumulating error. For example, by running the GPS sensor at a low-duty cycle [38]. Such systems can help reduce the GPS power consumption and even improve accuracy when the resetting frequency is small. However, when the GPS synchronization is more than a minute, these systems can suffer from localization errors as large as hundreds of meters. To further reduce the battery consumption and even reduce the GPS dependency for error resetting, the Dejavu [39] system leverages the array of independent sensors available on the phone for error resetting, achieving accuracies better than the GPS in some cases.

AccessMap is independent from the technology used to estimate the sensor measurements' location tag. Thus, GPS or other surveyed technologies can be used to tag the measurements. Yet, all these systems only focus on outdoor localization refinement and fail indoors. Thus, a different system is required to estimate the location tags indoors.

2.5 Smartphone-based Indoor Localization Systems

WiFi presents an attractive indoor localization technology due to its infrastructure world-wide deployment in all buildings and WiFi chip availability in all smartphones for internet connectivity. A GPS-similar approach for WiFi-based indoor localization is through triangulation. As the smartphone receives signals from its surrounding accesspoints, we can estimate the distance between the device and each accesspoint based on its received signal strength. Then, using triangulation, we can estimate the device location [40]. However, due to the wireless signal propagation sensitivity to noise in typical multipath-rich indoor environments [41, 42], triangulation leads to high localization errors. Thus, researchers proposed an offline calibration phase where the WiFi characteristics at the different locations in the area of interest is saved in a fingerprint map. Then, for localization, the map is compared to the user's real-time received signals (from the surrounding accesspoints) and the user is localized to the location of the most similar fingerprint [43, 44]. Nevertheless, conducting site-surveys, to construct and maintain the fingerprint map for every deployment area, increases the system's deployment overhead. Crowd-sourcing based

systems were introduced to reduce this overhead along with fusing WiFi with other sensors such as camera, accelerometer, gyroscope, etc... [42]. For example, in [45,46] authors crowd-source the smartphone's WiFi and inertial sensors to build a map of landmarks including stairs, spots with unique magnetic fluctuations or corners with unique WiFi signatures to localize users. In [47], the iMoon system crowdsources 2D photos to build 3D models of the environment and uses WiFi fingerprints to speed its image-based localization. Using additional sensors can limit the ubiquity and increase the power consumption of the system. Also, theoretical based approaches were presented to construct the fingerprint map automatically, e.g. AROMA [48,49]. However, the dependency in theoretical ray tracing leads to significant degradation in the localization accuracy.

Nevertheless, all these systems rely on the APs' RSS absolute value for localization. Recently, it was shown that the RSS absolute values are sensitive to heterogeneity. Thus, in [50–52], authors propose to use the APs relative order instead which improves the localization system resilience to heterogeneity. In [50] authors proposed to fingerprint the APs' RSS rank. This approach was extended in [51] and IncVoronoi [52] to remove the fingerprint map calibration requirements. In [51], authors assume a universal propagation model of the APs' signals and use that to construct a map of the APs' sequence based on their predicted RSS value. Then, in real-time, the user is located based on her visible APs' sequence similarity. Alternatively, instead of ordering the visible APs sequence, IncVoronoi [52] employs a pairwise comparison of the user's visible APs to identify her most probable location through Voronoi tessellation.

A prominent requirement for many location-based services, such as passive crowdsourcing systems, is knowing where exactly the user is located in the building—including which floor number. Thus, recently, WiFi-based localization systems have been proposed to identify the user's 2.5D location. In [53], authors propose to fingerprint the entire building. However, this increases the deployment and maintenance overhead significantly, particularly in buildings with a large number of floor-levels. ViFi [54] uses the Multi-Wall Multi-Floor propagation model [55] to automatically predict the APs' signal propagation and construct the multifloor fingerprint maps. However, theoretical models accuracy degrades significantly in uncontrolled environments [52, 54]. Contrarily, Locus [56, 57] is calibration-free system which uses a heuristics-based algorithm to identify the user's floor. Then, employs weighted triangulation for *APs from the estimated floor only* to estimate her 2D location. This approach trades accuracy for ease of deployment. TrueStory [58] is also a calibration-free system that estimates the user floor using a Multilayer Perceptron. However, it only uses the APs' floor number and ignores the APs 2D location within the floor. This can limit the system accuracy. Additionally, these systems were tested in low-rise ($\sim$4 floors) buildings only.

There have been some recent efforts to boost WiFi-based indoor localization systems' accuracy using Channel State Information (CSI) instead of the APs' RSS [59]. For example, the SpotFi [60] and the PhaseFi [61] systems which employed angle of arrival and fingerprinting based approaches respectively. Both systems were tested using the Intel 5300 WiFi NIC in a single-floor testbed area. CSI-based methods are not suitable for smartphone-based localization due to the unavailability of

the physical layer information from the smartphones' NICs and the mobile operating systems' APIs.

To address the aforementioned limitations, in this dissertation, we present the *Hapi* system which estimates the user 2.5D location in calibration-free manner. It identifies the user's floor using a scalable and general deep learning based method. Then, it refrains from using the APs' RSS absolute value when estimating the user's 2D location. However, instead of solely relying on the APs order, as discussed earlier, which totally ignores the APs' RSS levels, we propose an RSS-Rank Gaussian-based model that balances heterogeneity resilience and accuracy. Moreover, *Hapi* uses APs from all floors to identify the user 2.5D location, allowing it to work in realistic multistory buildings. The *Hapi* system was thoroughly tested in high-rise multistory buildings.

Chapter 3: Indoor Localization

3.1 Overview

Location is one of the most valuable user-context information, with a rapidly growing number of Location-based Services (LBSs) becoming an integral part of our daily life [3, 4, 19, 62–64]. Nevertheless, we are still missing a ubiquitous indoor localization system [65]. The world-wide ubiquity of WiFi networks, attracted lots of attention from researchers to utilize the already available WiFi infrastructure for identifying the user's indoor location [43, 45, 46, 51, 52, 57, 66]. Yet, currently available systems entail a tedious calibration/training phase for each deployment building/floor, require additional sensors, have a coarse-grained accuracy and/or sensitive to device/environment heterogeneity, among others. Another major additional limitation for the vast majority of available systems, e.g. [43, 45, 46, 51, 52, 66], is focusing on a single-floor area of interest which reduces the usefulness of their estimated location in realistic multistory buildings scenarios [57].

In this chapter, we present the *Hapi* system [67] as a novel WiFi-based 2.5D indoor localization system; i.e. it identifies the user's floor-level and her 2D location on that floor. It is calibration-free, **only requiring a building's floorplans and its WiFi APs' locations for deployment**. *Hapi*, instead of the typical approach

of estimating the user location in a 2D setting, works in a 2.5D setting incrementally. Our observation here is that while a user gets to hear faraway APs from her floor, she is less likely to hear far APs from other floors; as their signals get attenuated by the ceilings/floors as well as the 3D distance between the AP and the user. Figure 3.1, below, shows an example for a user standing in floor 4 and her visible APs from the different floors in the building (visible APs are marked with a tick). The user gets to hear more distant APs from the current floor (floor 4) as opposed to distant floors where she gets to only receive packets from the APs above her directly. We can see in the figure, the more distant the floor-number, the closer its visible APs to the user from a plan-view. Additionally, the user visible APs are within few floors above and below. Thus, no APs are visible to the user from the 1st and 7th floors.

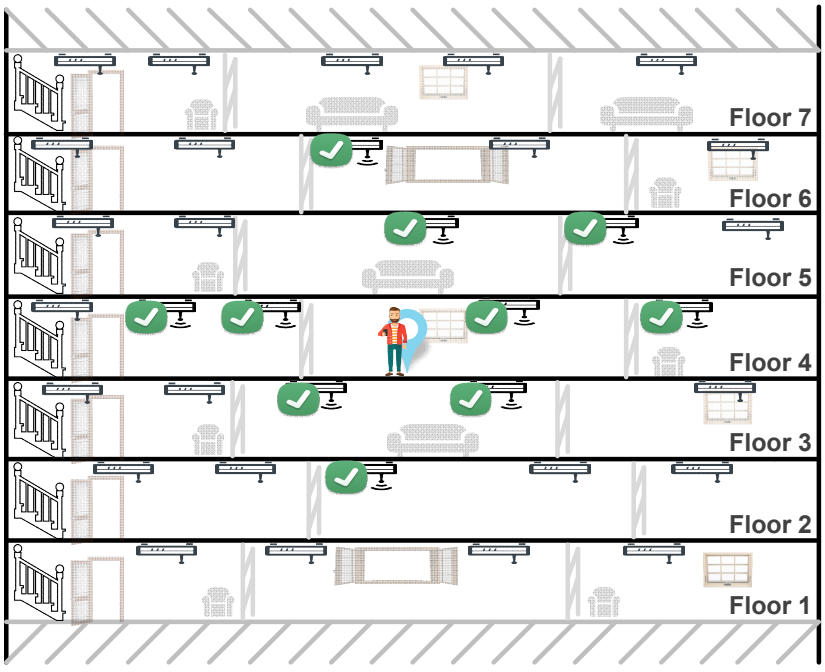


Figure 3.1: A user standing at floor 4 has visible APs (marked with a tick) from nearby floors along with her actual one (from floor 2 to 6). However, the user is hearing more distant APs from her floor (floor 4) as compared to APs from other floors.

Hapi takes this observation into account to improve the user location estimation on her floor. Thus, it starts by identifying the user's floor-level in the building. Next, that floor-level is used, along with the user's visible APs from all floors, in estimating her 2D location. In addition, we propose a regression-based algorithm to predict the quality of the estimated location and employ it within a Kalman-Filter (KF) to further refine *Hapi*'s location accuracy.

To identify the user floor, we propose *a novel deep-learning based method* that takes the user visible APs as an input and estimates the user's floor. More specifically, we start by limiting the floor search space to the set of floors where the user is most likely located. *This makes our method independent of the building's number of floors.* Then, we extract a set of features based on the APs' 2.5D installation location, such as the number of APs per floor and the farthest distance between visible APs per floor. Due to the WiFi signal propagation characteristics, typically, a user can hear more APs from her floor as compared to other ones and, as highlighted earlier, she can hear more distant APs from her floor as compared to other floors (Figure 3.1). The extracted WiFi features are then fed into the deep-network for floor prediction. Additionally, to train the network, we present a data balancing and generalization approach that improves the system accuracy and robustness.

Thereafter, to identify the user 2D-location on her floor, *Hapi* considers the user's *visible APs from all floors* and employs her floor-level to compensate for the attenuation the APs' RSS incurs (due to propagation through floors/ceilings). Next, the processed RSS is used to estimate the user location PDF over her entire floor. However, to make the location estimation robust to heterogeneity, *Hapi* refrains from

using the absolute APs' RSS values while estimating the PDF. Instead, building on the work of [51, 52], we define RSS-Rank that ranks APs' RSS to levels e.g. strong, weak, etc... and use the RSS-Rank in a Gaussian-based method to assess the user proximity probability from each AP across the floor. This helps reduce the effect of the RSS variability, as while the APs' RSS varies (e.g. among devices), their Rank is more robust. For instance, considering a distant AP with a weak RSS, while its absolute RSS may vary, its RSS-Rank is not likely to change from weak to strong.

We have deployed *Hapi* on different android devices (covering a wide-range of models including Samsung, Motorola, LG, OnePlus and Huawei) and conducted extensive experiments *spanning 6 months with 13 subjects in five different multistory buildings with up to 9 floor levels*. We present our implementation and evaluation in Chapter 4, as well as a comparison with state-of-the-art indoor localization systems.

The rest of this chapter is organized as follows: Section 3.2 presents an overview of the *Hapi* system architecture. Then, we give the details of its floor-level and 2D location estimation in Section 3.3 and 3.4 respectively. Finally, we summarize the chapter in Section 3.5.

3.2 The Hapi System Architecture

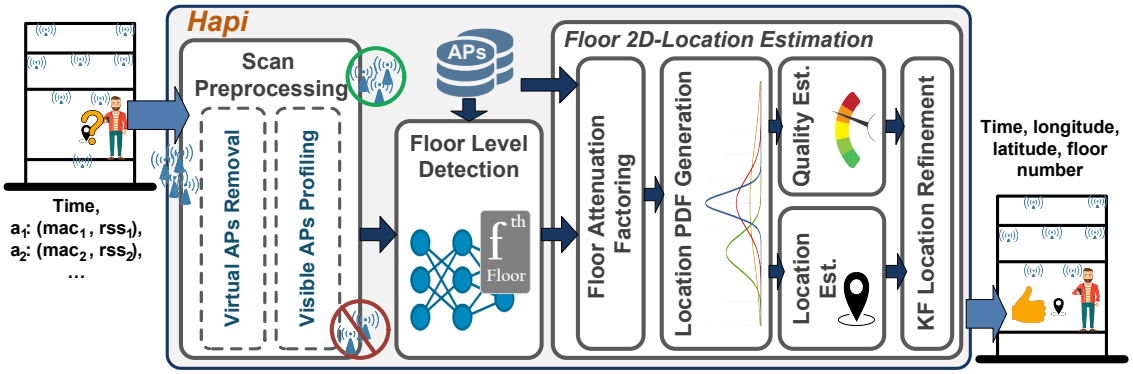


Figure 3.2: The *Hapi* system architecture. *Hapi* takes only the user's visible APs and their RSS to estimate her 2.5D location (the floor number and the longitude and latitude of her 2D-location on that floor).

Figure 3.2, above, shows the *Hapi* system architecture. To deploy *Hapi* in a building, only a list of the building's installed WiFi APs' MAC addresses and installation locations is fed to the system along with the building's floor plans. To localize a user, the system takes a time-stamped WiFi scan list which consists of the user's visible APs' MAC addresses and their received signal strengths (RSSs). *Hapi* has three main modules: the *Scan Preprocessing* module, the *Floor Level Detection* module and the *Floor 2D-Location Estimation* module.

3.2.1 Scan Preprocessing Module

For a given WiFi scan list, we start by mapping any virtual AP to its physical AP MAC address. A virtual AP is a multiplexed installation of a single physical AP so that it presents itself as multiple discrete APs to the wireless LAN clients. The mapping function should be provided at deployment with the building's WiFi

network information. For example, in our testbed buildings, the last hexadecimal digit for a physical AP's MAC address is 0 and has different values for its virtual APs. Thus, we mask the last hexadecimal digit to 0 to map all virtual APs to their physical AP's MAC address. Then, we set the physical AP's RSS to the average RSS of its virtual APs.

In addition, *Hapi* creates a profile for the user's surrounding APs and their RSS through using a sliding window of the WiFi scans. This is intuitive as within a short period, in typical indoor scenarios, the user is less likely to move far from her current location and the window can show more APs in the user's vicinity—a single WiFi scan may only show a subset of the nearby APs. Thus, for a window of size N, a WiFi profile (p_t^N) of the user's surrounding WiFi network at time t is constructed by taking the union of all visible APs in N scans. If an AP is visible in multiple scans, its strongest RSS is used in p_t^N. More formally, p_t^N is defined as follows:

$$\begin{aligned} p_t^N &= \bigcup_{i\,=\,max(0,t-N)}^{t} s_i \\ &= \{a_m : a_m = (mac_m, rss_m), \exists s_i \supset a_m, i \in [max(0, t-N) : t]\} \end{aligned} \tag{3.1}$$

Where $s_i = \{a_1, ..., a_m, ...\}$ is the WiFi scan list at time i with an AP $a_m = (mac_m, rss_m)$ is defined by its MAC (mac_m) and RSS (rss_m).

3.2.2 Floor Level Detection Module

This module is responsible for detecting the user's floor level. It takes as an input, a visible APs profile $p_t^{N_f}$. We show the N_f size effect in Chapter 4. The module extracts a set of features, from $p_t^{N_f}$, that depict the WiFi characteristics of each floor using the floor's APs locations. For example, it extracts the number of APs from every floor. The intuition for this feature is that, a user is more likely to have a larger number of visible APs from her floor as opposed to other floors, due to signal attenuation (e.g. Figure 3.1). Then, the extracted features are used to estimate the user floor f_t through a deep neural network. Furthermore, the module incorporates RSS variability for data balancing and generalization while training the network to achieve high and robust floor estimation accuracy. The operation details of this module are discussed in Section 3.3.

3.2.3 Floor 2D-Location Estimation Module

This module is responsible for estimating the user 2D location within her floor level. It takes as an input, a visible APs profile $p_t^{N_l}$ and the user's estimated floor level f_t. We show the N_l size effect in Chapter 4. *Hapi* estimates the user location using a novel probabilistic method as we discuss in detail in Section 3.4.

Note that the *Floor 2D-Location Estimation* module has $N = N_l$ which is different from the N_f used in the *Floor Level Detection* module. This is intuitive as the user will typically be moving/walking for a while in the same floor as compared to her location within that floor. Thus, we expect N_f to be longer than N_l.

3.3 Floor-Level Detection

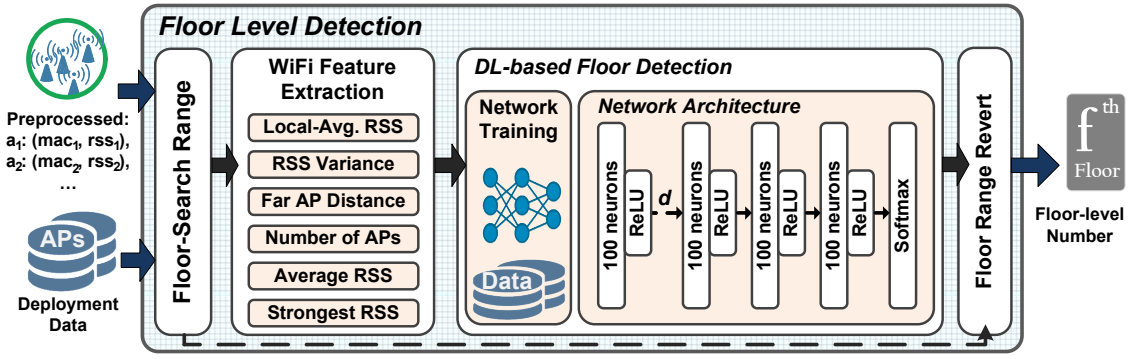


Figure 3.3: Overview of *Hapi*'s *Floor Level Detection* module.

A key part of *Hapi*'s localization algorithm is identifying the user's current floor. Figure 3.3 shows an overview of the *Floor-Level Detection* module. The *Floor-Level Detection* module takes $p_t^{N_f}$ and outputs the user's floor estimate f_t. The module starts by finding the floors' range where the user is most-likely located out of the entire building. Then, WiFi features are extracted and are fed to a Deep Neural Network (DNN) to identify the user's floor.

3.3.1 Floor-Search Range

While buildings can consist of any number of floors, WiFi APs have limited reception range. For example, Figure 3.4 shows a histogram of the difference between the floor where an AP is visible and its installation floor for all APs in our experiments data which includes up to 9 floor buildings (Chapter 4).

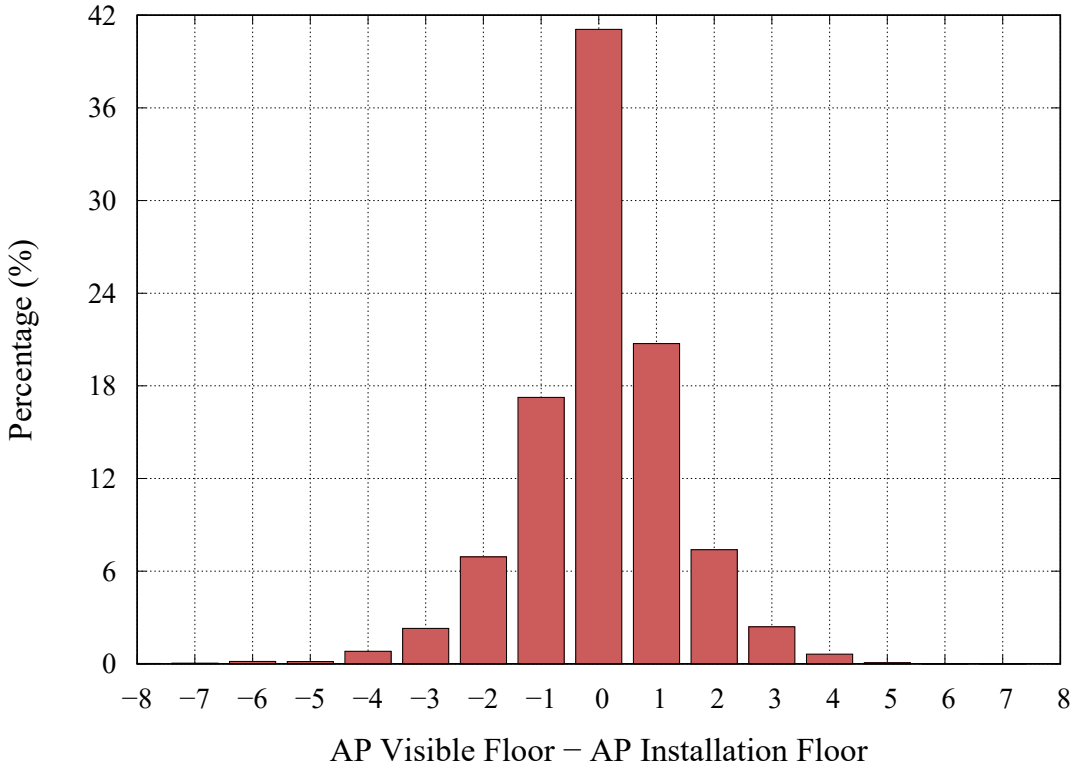


Figure 3.4: Histogram of the difference between the floor where an AP is visible and its installation floor for APs in our experimental data.

We can see that typically, 98% of the time, an AP can only be heard within four floors from its installation floor. We leverage this to limit the search range of the user-floor finding algorithm. Specifically, for a user with WiFi profile p_t^N in a building with $\mathcal{F}$ floors, *Hapi* limits its floor-search range to a subset $F \in \mathcal{F}$ of up to w consecutive floors. w is a system parameter that defines maximum number of floors where we search for the user location. Thus, $|F| \leq |\mathcal{F}|$ and $|F| \leq w$. This enables our floor-finding algorithm to work on w floors, independently from the building's number of floors $|\mathcal{F}|$, **allowing *Hapi* to be deployable in any building with any number of floors**. Due to signal propagation characteristics, APs closer to the user will have stronger RSS. Thus, we define F as the consecutive

subset of floors where the user is covered with the strongest overall RSS. The search range F start index r_1 is computed as follows:

$$r_1 = \underset{f \in [1:|\mathcal{F}-w|]}{\arg\max} \sum_{i=f}^{f+w} \sum_{\forall a_m \in A_i} rss_m \tag{3.2}$$

Where A_i is the set of APs installed in floor i. Note that, APs from floors outside F (if any) is removed as they represent outlier weak APs. We set w to 4 based on our analysis (Figure 3.4) and evaluation.

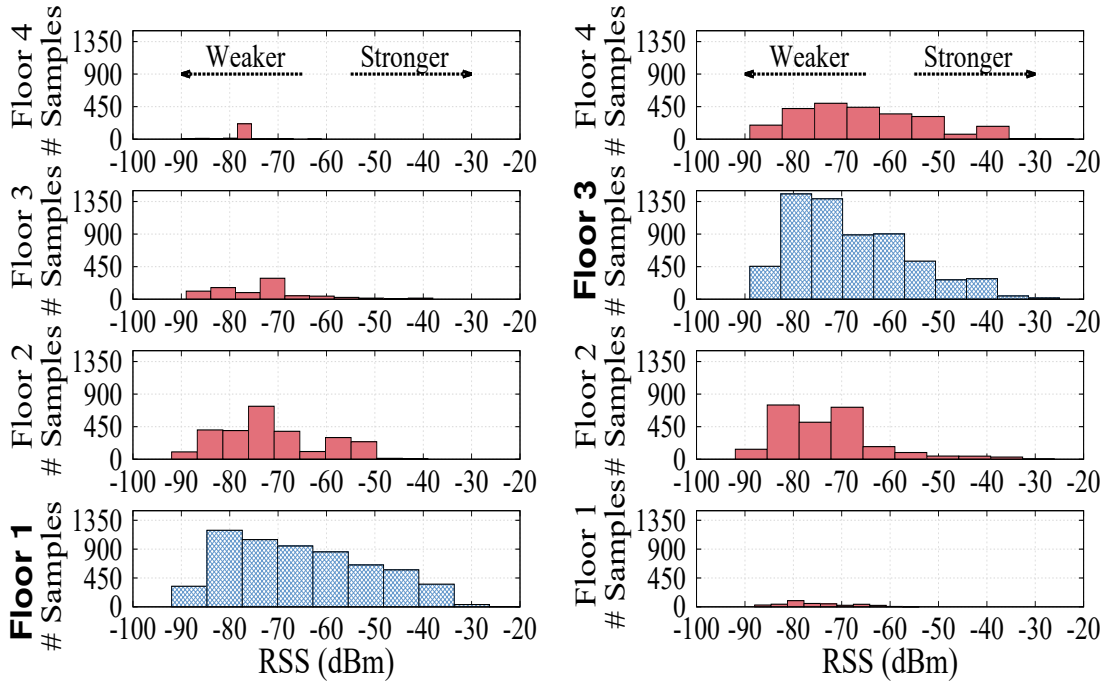
3.3.2 WiFi Feature Extraction

For a WiFi profile p_t^N, we extract a set of features from each of the w floors. First, we extract features based on the building's APs' installation floor: The number of access-points, the strongest signal strength, the average signal-strength and the signal strength variance. In addition, we propose features that are based on the APs' 2.5D installation location: The local-average signal strength and the farthest access-point distance per floor. We describe each in detail next.

3.3.2.1 Number of Access-points

WiFi accesspoints have a limited range (around ∼100m) and this range decreases with attenuation from walls and other indoor clutter. Thus, we expect to have more visible APs from the user current floor and the visible APs number should decrease as we move away from the user's floor (Figure 3.1). For instance, Figure 3.5 shows the histogram of visible APs' RSS in each floor, within the search range, for

users walking in 1st and 3rd floors of a building. We can see that more APs were visible in the users' floors (1st and 3rd respectively).



(a) User location: **1st floor** (b) User location: **3rd floor**

Figure 3.5: Histogram of visible APs' RSS for users walking in the 1st floor vs 3rd floor of the testbed Bldg. 115.

Therefore, our first extracted feature is the number of visible APs per floor (Num_f) in the WiFi profile p_t^N. To Put it, formally, for each floor $f \in F$:

$$Num_f = |A_f| \tag{3.3}$$

3.3.2.2 Strongest Signal Strength

WiFi signals get attenuated, by distance and obstacles (e.g. floors/ceilings), leading to weaker RSS. Thus, we expect APs in the user's floor to have a relatively strong RSS compared to APs in distant floors. Going back to the visible APs' RSS

histogram example in Figure 3.5, we can see that there is a relatively high number of APs with strong RSS (> -60dBm) in the user's floor (1st and 3rd respectively) and this number decreases as we go up/down. Thus, our second extracted feature is the strongest RSS per floor (Str_f) in the WiFi profile p_t^N. To Put it, formally, for each floor $f \in F$:

$$Str_f = \max_{a_m \in A_f} rss_m \tag{3.4}$$

3.3.2.3 Average Signal Strength

Other than decreasing the maximum RSS, all signals coming from APs in distant floors get attenuated, hence, their average RSS decreases as well. We can see in Figure 3.5 that the user's floor has higher ratio of APs with strong RSS (> -60dBm) thus it can have higher visible APs' RSS on average. Therefore, the next feature we extract is the average visible APs' RSS per floor (Avg_f) in the WiFi profile p_t^N. To Put it, formally, for each floor $f \in F$:

$$Avg_f = \frac{1}{|A_f|} \sum_{a_m \in A_f} rss_m \tag{3.5}$$

3.3.2.4 Signal Strength Variance

While all signals from APs' installed at distant floors get attenuated by the ceilings/floors between the APs and the user, this is not the case for APs in the user's floor. APs installed at the user's floor, based on their proximity to the user, can have very strong or weak RSS. For instance, in Figure 3.5, whereas APs at distant floors

exhibit a predominantly weak RSS, the user-floor's APs have both weak and strong RSS. Leading to a higher RSS variance in the user's floor as compared to distant ones. Hence, we extract the RSS variance per floor (Var_f) in the WiFi profile p_t^N. To Put it, formally, for each floor $f \in F$:

$$Var_f = \frac{1}{|A_f|} \sum_{a_m \in A_f} (rss_m - Avg_f)^2 \tag{3.6}$$

3.3.2.5 Local-Average Signal Strength

One problem with taking the average of all visible APs' signal strength is that, the user's actual floor can have high RSS variance (as discussed in previous features), making the average not representative. To overcome this problem, we propose the *local-average signal strength* feature which utilizes the installed APs' 2.5D location. Particularly, rather than computing the visible APs' average RSS within the entire floor, we compute a local-average RSS for all installed APs within a *proximity region* (Figure 3.6).

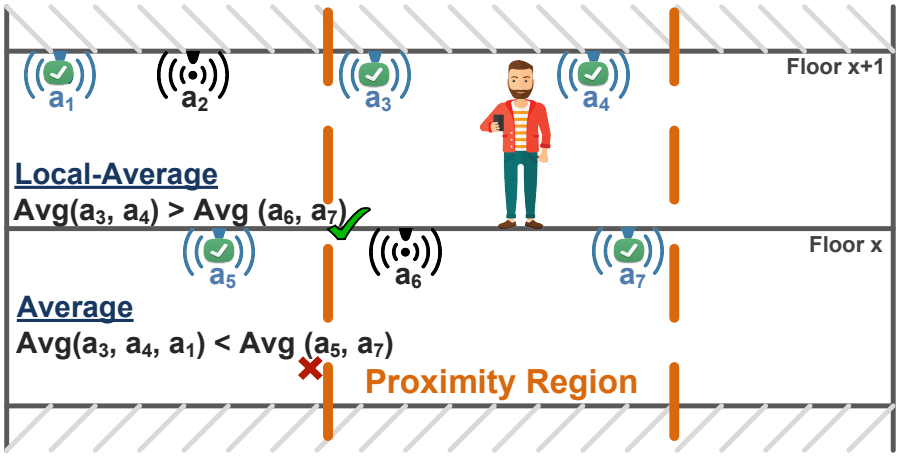


Figure 3.6: Example illustrating Local-Average Signal Strength and its proximity region vs. Average Signal Strength computation. Visible APs are marked with a tick.

If an AP in the region is not visible, we assume a weak RSS of $-100dBm$. Figure 3.7 compares the extracted *Average Signal Strength* and *Local-Average Signal Strength* features and shows how using the local-average can reduce the user's floor high RSS variance problem through taking the WiFi network deployment into account.

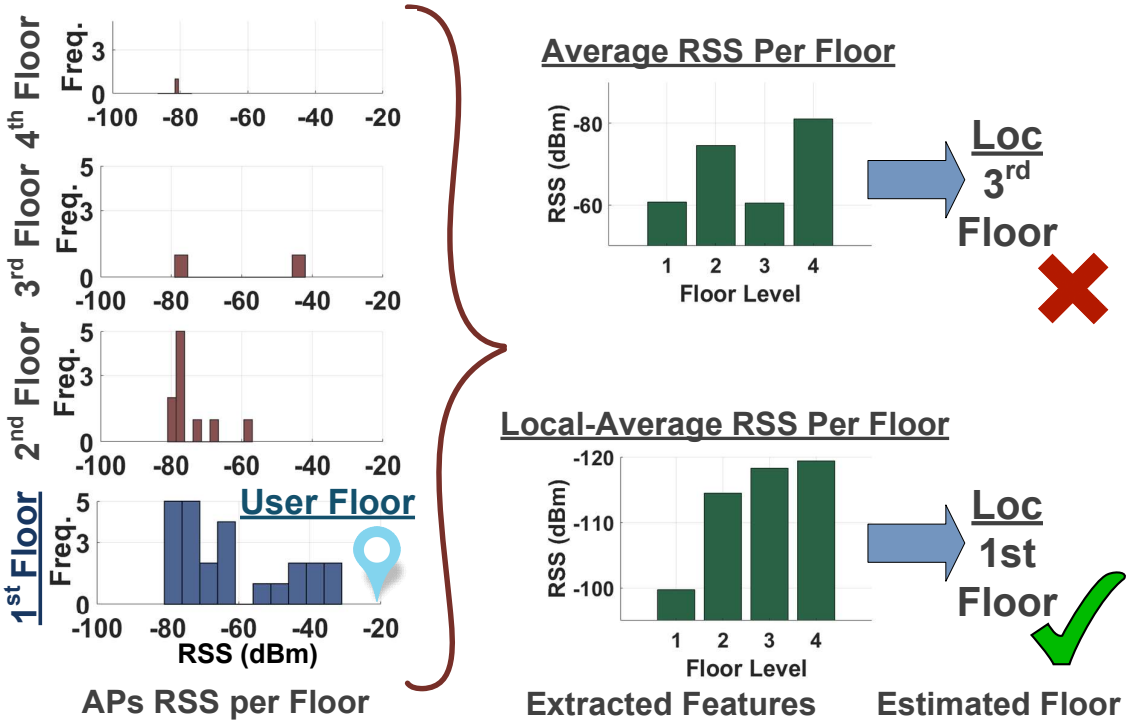


Figure 3.7: Example comparing the *Average Signal Strength* to the *Local-Average Signal Strength* features on user-floor detection.

The proximity region $\mathcal{R}_\delta$ is defined by triangulating the WiFi profile p_t^N APs' 2D locations and drawing a circle with radius δ around it. Then, for each floor $f \in F$, the local-average RSS ($LocAvg_f^\delta$) is computed as follows:

$$LocAvg_f^\delta = \frac{1}{|I_f^\delta|} \sum_{a_m \in I_f^\delta} rss_m \tag{3.7}$$

Where I_f^δ is the set of APs at floor f in $\mathcal{R}_\delta$. We set δ to 30m, 80m and the farthest distance between visible APs α (Equation 3.8) to cover different regions granularity.

$$\alpha = \max_{a_m \in p_t^N, a_l \in p_t^N} dist(a_m, a_l) \tag{3.8}$$

Where $dist(a_m, a_l)$ is the Haversine distance between a_m and a_l. We analyze the effect of δ in Chapter 4.

3.3.2.6 Farthest Access-point Distance

As discussed earlier, the user is more likely to hear distant APs from her floor as compared to other floors (e.g. Figure 3.1). Thus, to capture this observation, we extract the farthest pairwise visible APs distance (*Far*) at each floor as our last feature. For each floor $f \in F$:

$$Far_f = \max_{a_m \in A_f, a_l \in A_f} dist(a_m, a_l) \tag{3.9}$$

3.3.3 Deep Learning-based Floor Detection

We formulate the floor detection problem as finding which one of the w search-range floors is the user's actual floor ($f_t{}'$) and propose a Deep Neural Network (DNN) to solve it. The extracted WiFi features from each of the w floors are fed into the DNN and it outputs $f_t{}'$. Our DNN has four fully connected layers each with 100 neurons. Additionally, we design our network architecture to include the following: (1) a dropout layer with a dropout ratio of 0.25 (25% chance of setting a neuron's

output value to zero) after the first fully connected layer [68] to improve the network generalization capability; and (2) we set the non-linearity for the four fully connected layers to be fulfilled by Rectified Linear Unit (ReLU) [69] as the activation function. ReLUs help prevent saturation of the gradient when the network is deep. The network ends with a softmax layer to predict the floor-level probability of each of the search-range w floors and $f_t{}'$ is set as the most probable one. A basic layer block is formalized as:

$$y = \mathbf{W} \cdot \mathbf{x} + \mathbf{b} \tag{3.10}$$

$$h = ReLU(y) \tag{3.11}$$

Where $\mathbf{W}$ denotes the weights vector, $\mathbf{x}$ is the features vector, $\mathbf{b}$ is the bias. In the layer where we add the dropout (with dropout ratio d), $\mathbf{x}$ is substituted by $\tilde{\mathbf{x}}$ where:

$$\tilde{\mathbf{x}} = f_{dropout,d}(\mathbf{x}) \tag{3.12}$$

In the first layer, $\mathbf{x}$ is the extracted WiFi features vector. Figure 3.3 shows the neural network architecture.

3.3.3.1 Training: Data Generalization and Augmentation

To deploy *Hapi*'s Floor-level Detection module in a building, no WiFi data collection is required from that building. Instead, to train the network, we emulate the deployment-building's WiFi network. In addition, we generalize and balance the training data's WiFi scans to prevent overfitting. Specifically, for each training data

instance, we start by sampling the APs per floor to match the number of Aps per floor in any randomly selected consecutive w floors (search-range) of the deployment-building. Then, we perturb the sampled-APs' RSS by adding a Gaussian random amount with zero mean and standard-deviation σ proportional to the degree of noiseness we want to add to the model. Based on our analysis of the APs' RSS variability, we set σ to 1.5. Finally, to balance the data, we repeat the Gaussina-random perturbation to equalize the number of instances in each floor. We evaluate the effect of our training methodology in Chapter 4.

3.3.3.2 Training: Optimization

We use a categorical cross entropy loss function and the Adam optimization algorithm ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, a batch size of 10 and a learning rate of 10^{-3}) [70] for training the network.

3.3.4 Floor-Range Revert

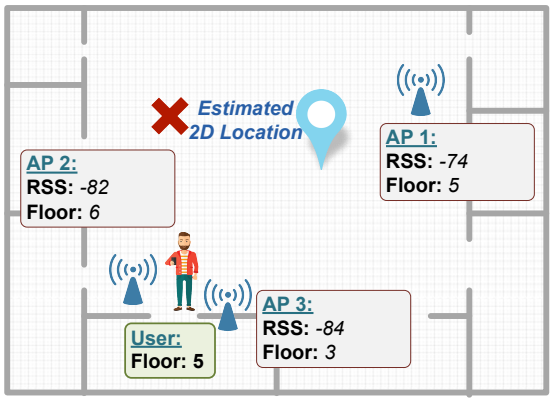
The DNN estimates the user's most probable floor $f_t{}'$ out of the search-range w floors. Then, we revert $f_t{}'$ back to its actual floor-level number f_t out of the building's $\mathcal{F}$ floors using the search-range 1st index r_1 (Equation 3.2): $f_t = f_t{}' + r_1$.

3.4 Floor 2D-Location Estimation

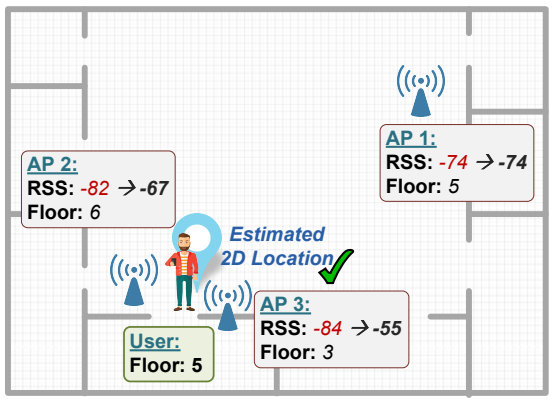
In this section, we provide *Hapi*'s *Floor 2D Location Estimation* module details. It takes $p_t^{N_l}$ along with the user's detected floor f_t, and outputs her 2D location

on floor f_t. The module starts by preprpocessing the APs to compensate for the attenuation their RSS incur due to their installation floor. Then, we employ a novel RSS-Rank Gaussian model to estimate the user location PDF across her entire floor and assess the user's most probable location. Finally, a regression-based model is proposed to predict the quality of the user-location and it is employed within a KF to further refine that location.

3.4.1 Floor Attenuation Factoring



(a) **Without** Floor Attenuation Factoring



(b) **With** Floor Attenuation Factoring

Figure 3.8: Example showing the effect of the APs' installation floor on their RSS and *Hapi*'s 2D location estimation with and without the *Floor Attenuation Factoring* module. The user is located in floor 5.

While users typically have visible APs from a number of floors nearby her actual floor, those visible APs' signals get attenuated by the floors/ceilings in between. Although this attenuation helps in identifying the user floor-level (as described in Section 3.3), it leads to confusion and significant errors when estimating the user 2D location on her floor (e.g. Figure 3.8(a)). To address this issue, we build on the well-known Wall Attenuation Factoring method, which have been used in indoor localization systems e.g. [40,52], and compensate for the APs' installation floor number through the Floor Attenuation Factoring [71] (Figure 3.8(b)). *Hapi* leverages the user floor estimate f_t and the APs installation floors location to estimate the number of ceilings/floors between the user and each of her visible APs. In particular, for a user at floor f_t with a WiFi visible APs profile p_t^N, the received signal strength of every AP $a_m \in p_t^N$ is updated as follows:

$$rss_m = rss_m + W_f \times |(f_t - f_m)| \tag{3.13}$$

Where f_m is the AP a_m installation floor and W_f is a constant parameter representing the signal attenuation due to floors/ceilings. We evaluate the effect of W_f on *Hapi*'s performance in Chapter 4.

3.4.2 Location PDF Generation

Next, using the processed WiFi profile p_t^N, *Hapi* estimates the PDF of the user location (denoted by $\mathcal{L}(p_t^N, \ell_{(x,y)})$) over her entire floor using a novel RSS-Rank Gaussian-based method. $\ell_{(x,y)}$ is the longitude and latitude location variable. A

key observation noted in related work is that using the exact RSS value to estimate the user-location leads to device and environment heterogeneity problems and can limit the ubiquity of the proposed model [52]. To address this issue, related work systems relied on the relative order between the APs such as IncVoronoi [52] and Liu et al. [51]. However, while this approach helps with the heterogeneity and model overfitting, it has two main issues: First, when APs are having close RSS values (e.g. -76 and -77), ordering them or saying that you are closer to any of them is dubious. Second, when using the relative order, you are losing part of the information in p_t^N (i.e. the APs' signal strength level). Thus, *Hapi* extends on related work and normalizes the RSS values to RSS-Ranks. RSS-Ranks rates APs based on their RSS to different ranks such as strong, moderate and weak. Using ranks gives the benefits of the APs ordering (we are not using the RSS absolute value). Yet, we are still relatively using its strength level through the normalization ranks.

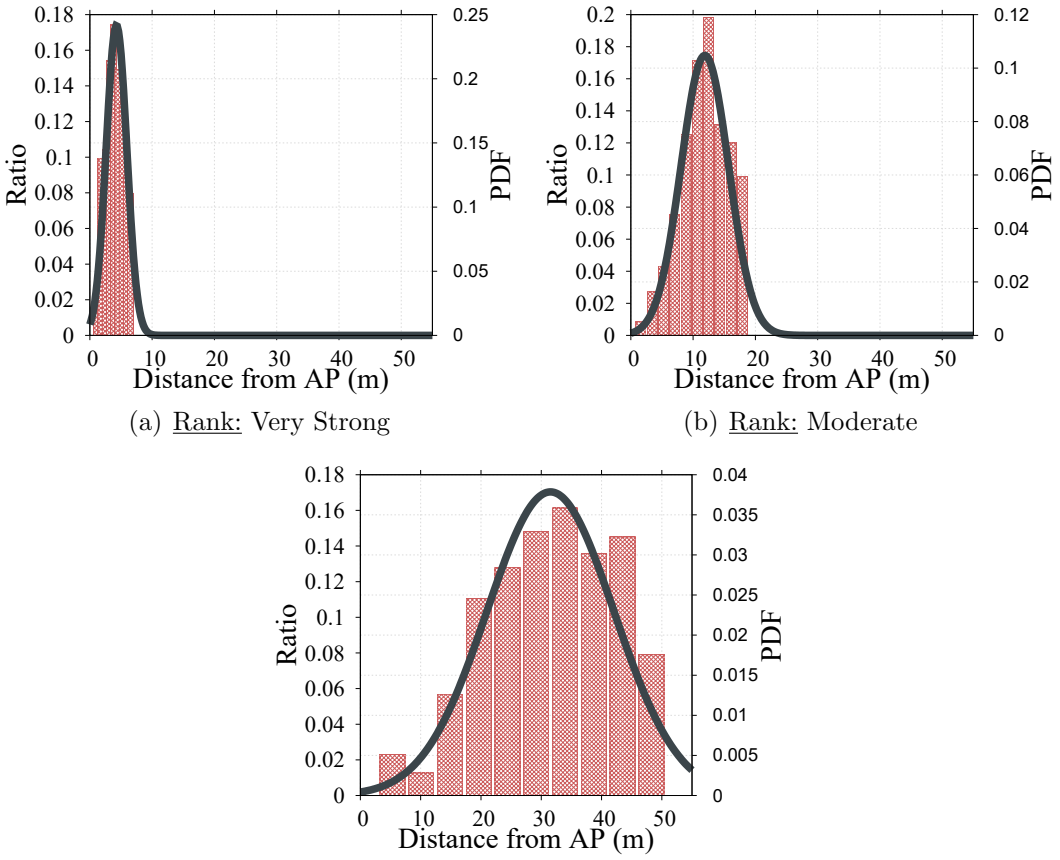
To estimate the user-location PDF $\mathcal{L}(p_t^N, \ell_{(x,y)})$ (Equation 3.14), we model the probability of being at distance d from an AP with RSS-Rank z using a Gaussian distribution $\mathcal{N}(\mu_z, \sigma_z)$.

$$\mathcal{L}(p_t^N, \ell_{(x,y)}) = \sum_{a_m \in p_t^N} \mathcal{N}(dist(a_m, \ell_{(x,y)}), \mu_z, \sigma_z) \tag{3.14}$$

Where $\mathcal{L}(p_t^N, \ell_{(x,y)})$ is the probability of the user 2D location $\ell_{(x,y)}$ at time t, z is AP a_m RSS-Rank and $dist(a_m, \ell_{(x,y)})$ is the Haversine distance between AP a_m and 2D-location $\ell_{(x,y)}$.

To verify the Gaussian-based model and estimate its parameters, we conducted

an experiment in testbed Bldg. 115 4th floor (Chapter 4). The RSS values were clustered based on their APs' Haversine distance to the user location using hierarchical clustering, yielding the following ranks: very weak (RSS > -80), weak ($-70 >$ RSS > -80), mild ($-60 >$ RSS > -70), moderate ($-60 >$ RSS > -50), strong ($-50 >$ RSS > -40) and very strong ($-40 >$ RSS). Each cluster distances were then fitted to a Gaussian distribution after removing the outilers/anomalies [39]. We empirically choose 0.8 as the anomalies threshold. Figure 3.9 shows examples of the Gaussian distributions from the different RSS-Ranks along with histogram of user's distances from APs with outliers removed for clarity.



(a) Rank: Very Strong

(b) Rank: Moderate

(c) Rank: Weak

Figure 3.9: Histogram of user's distance from visible APs with different RSS-Ranks (very strong, moderate, weak) collected in testbed Bldg. 115 and *Hapi*'s RSS-Ranks' Gaussian models' PDF.

Note that, the RSS-Rank Gaussian model is used to estimate the location PDF in all evaluation testbeds without any training or calibration data collection from the testbeds.

3.4.2.1 Location Estimation

Using the generated PDF, the user location at time t (loc_t) is estimated as the weighted centroid of the area where the user is located with probability more than or equal ℓ_{thr} (Equation 3.15). We evaluate the effect of the ℓ_{thr} value on *Hapi*'s accuracy in Chapter 4.

$$loc_t = \frac{\int_T \ell_{(x,y)} \mathcal{L}(p_t^N, \ell_{(x,y)})\ d\ell_{(x,y)}}{\int_T \mathcal{L}(p_t^N, \ell_{(x,y)})\ d\ell_{(x,y)}} \tag{3.15}$$

Where T is the area with $\mathcal{L}(p_t^N, \ell_{(x,y)}) \geq \ell_{thr}$.

3.4.3 Location Quality Estimation

Generally, localization systems' accuracy are experimentally validated to assess their typical performance under different scenarios using ground-truth data. Although this can help end-users get a sense of the system's expected performance, it fails to provide any clue about the real-time error in the system's estimated location coordinates. Therefore, availability of a quality measure can enhance the system usability for end-users. Moreover, it can help further refine the system's tracking performance as we show next (Section 3.4.4). Typically, the quality measure is represented as a circle of ambiguity with the estimated location at its center

and its radius is the predicted quality [72]. Figure 3.10 below shows an example for a user standing in room A and her location estimate. While the location estimate is in room B but the circle of ambiguity highlights that the user can be in room A also.

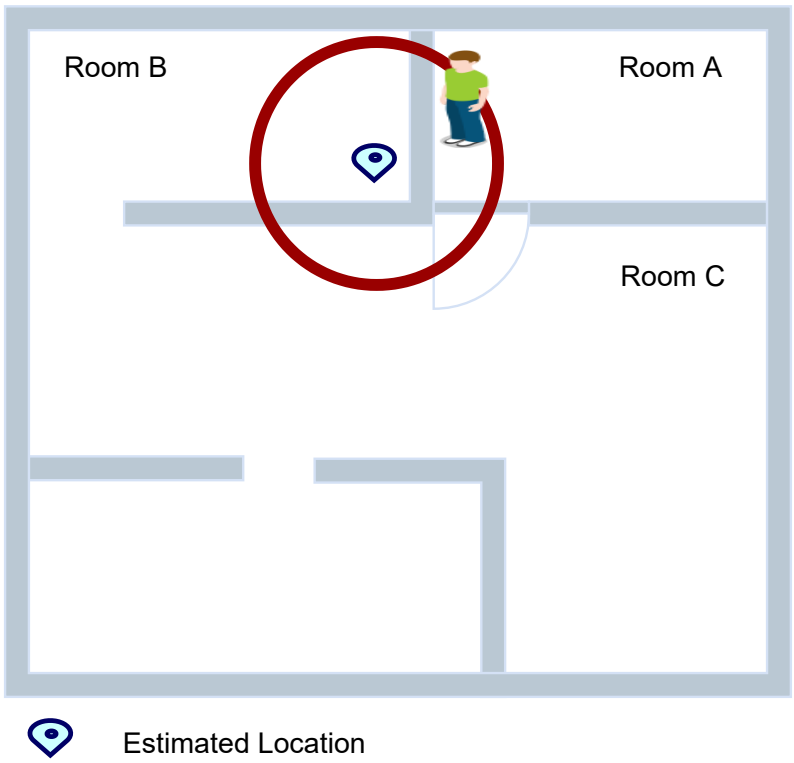


Figure 3.10: Example showing a user standing in room A, the location estimate and its circle of ambiguity.

To predict the quality of the user-location estimate, we analyze the different sources affecting the system's accuracy and model the relation between the accuracy and these sources using regression. Our analysis show that for a p_t^N, the following parameters affected the system performance: The visible APs number, the visible APs' average RSS, the maximum RSS and the area of the floor-space with $\mathcal{L}(p_t^N, \ell_{(x,y)}) \geq \ell_{thr}$, as it represents the uncertainty-level in the user-location PDF. Figure 3.11 shows an example comparing the actual error to the regression model

predicted quality. The quality metric has a high precision and recall for instances with high and low accuracy.

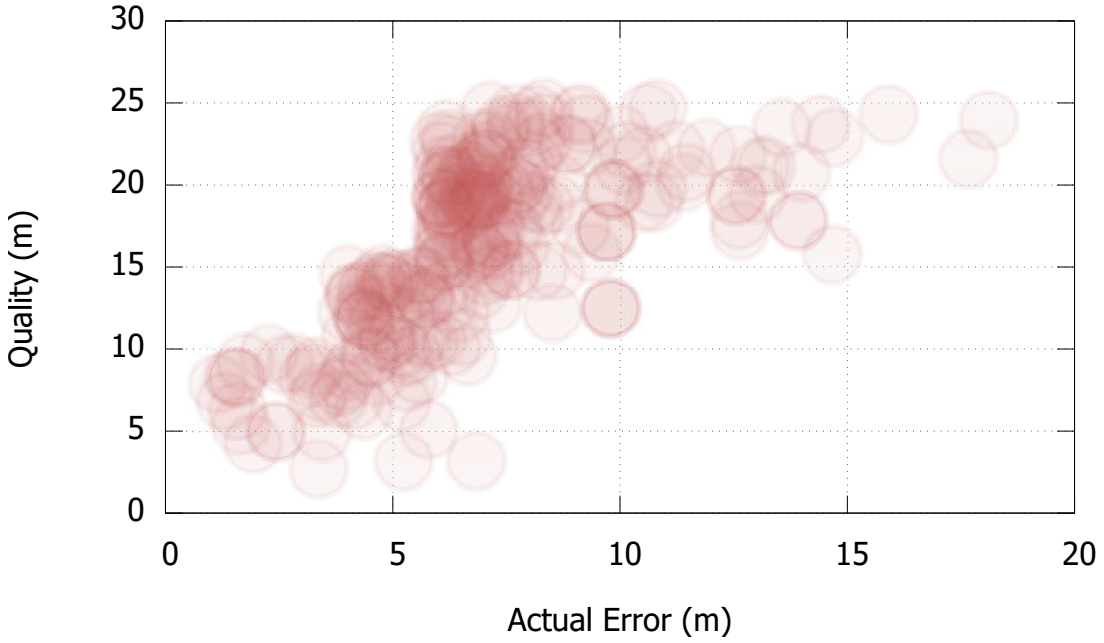


Figure 3.11: Example comparing the estimated-location actual-error and its predicted quality.

The regression weights are estimated using the same collected data in the 4th floor of testbed Bldg. 115 for the RSS-Rank Gaussian model validation.

3.4.4 KF Location Refinement

To refine *Hapi*'s location estimate and improve its tracking performance, we employ the user location history, as a window of successive location predictions and their estimated quality through a Kalman Filter (KF) [73]. We apply the KF on a window of size w_k and assume that the user is moving with a constant speed over that period. More specifically, we define our state ($\mathbf{s}$) as the user location coordinates ($\mathbf{x}$) and her speed components ($\mathbf{v}$): $\mathbf{s} = [\mathbf{x} \ \ \mathbf{v}]^T$. Thus, the state prediction at time

i (prior estimate: $\hat{\mathbf{s}}_i^-$) and its prior variance (P_i^-) are estimated as follows:

$$\hat{\mathbf{s}}_i^- = A_i \, \hat{\mathbf{s}}_{i-1} \tag{3.16}$$

$$P_i^- = A_i \, P_{i-1} \, A_i^T + Q_i \tag{3.17}$$

Where A_i is the state transition matrix at time step i, we model it as a constant speed linear motion:

$$A_i = \begin{bmatrix} 1 & \Delta t_i \\ 0 & 1 \end{bmatrix} \tag{3.18}$$

Q_i is the process noise at time step i which allows tracking of different forces that could affect the user's movement speed:

$$Q_i = \begin{bmatrix} \frac{\Delta t_i^2}{2} \\ \Delta t_i \end{bmatrix} \times \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \times \begin{bmatrix} \frac{\Delta t_i^2}{2} \\ \Delta t_i \end{bmatrix}^T \tag{3.19}$$

However, since we are doing the KF-refinement over a window rather than continuous tracking, a $\sigma_1 = \sigma_2 = 0$ gave us good accuracy as the user speed is less likely to change over the window. Next, using a new measurement, the KF updates the state as follows:

$$K_i = \frac{P_i^-}{P_i^- + R_i} \tag{3.20}$$

$$\hat{\mathbf{s}}_i = \hat{\mathbf{s}}_i^- + K_i(z_i - \hat{\mathbf{s}}_i^-) \tag{3.21}$$

$$P_i = (1 - K_i)P_i^- \tag{3.22}$$

Where at time step i, K_i is the Kalman gain, R_i is the measurement noise and is modeled using the *Location Quality* estimate described in Section 3.4.3, z_i is the user estimated location at time i and speed based on her location estimates at times i and $i-1$. Initial state is set as the user location and 0 speed with high variance (100). Empirically, we choose $w_k = 1$min as it balances accuracy refinement with the user location history and capturing her motion dynamics.

3.5 Summary

We presented the design of *Hapi*, a novel calibration-free WiFi-based indoor localization system that can be deployed easily in realistic multistory buildings. *Hapi* identifies the user's 2.5D location: her floor-level and 2D-location on that floor. This enables *Hapi* to give a detailed enough tag to indoor trajectories for passive crowdsourcing systems such as *AccessMap*. We described *Hapi*'s basic idea and showed how it combines a deep-learning based method to identify the user's floor, with an RSS-Rank Gaussian-based method to estimate the user's 2D location on that floor. Moreover, we present a regression-based method to predict *Hapi*'s location estimates' quality and employ it within a KF to further refine the location tracking performance.

We discuss *Hapi* implementation and evaluation in Chapter 4. Additionally, we compare it to related calibration-free state-of-the-art systems.

Chapter 4: Indoor Localization Performance

4.1 Methodology

We implemented *Hapi* using a client-server architecture where the client is an Android mobile app that collects the raw WiFi-scans through the Android API and sends them to the server. The localization algorithm is implemented as a web-service on the server. To evaluate the system accuracy under different WiFi-network characteristics, we used *five different testbeds* that exhibit different floor-plans, building structures and number of floors. *A total of 10423 samples were collected by 13 subjects (7 females and 6 males) within a period of 6 months.* The subjects used different android devices including LG Nexus 5X, LG Nexus 5, Motorola Droid Razr HD, Moto G4, Samsung Galaxy Tab 4, Samsung Galaxy J7, Samsung Galaxy J3, Samsung Galaxy S III, OnePlus 3 and Huawei Mate 9. To obtain the ground truth, we developed a special app using Google Indoor Maps [74], where subjects report their floor-level ground truth. Afterwards, the app shows them their current floorplan indoor maps and they mark their ground-truth 2D longitude and latitude location coordinates on the map as they walk. Thus, for every WiFi-scan the app reports a 2.5D ground-truth location. Figure 4.1 shows Bldg. 115 (one of the 5 testbeds) 3rd floor floorplan and a sample of the ground-truth user trajectories.

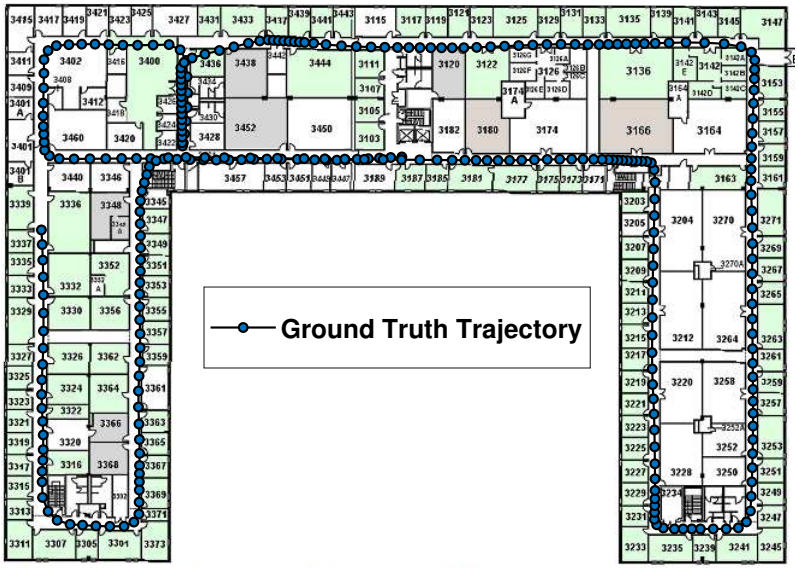


Figure 4.1: Bldg. 115 3rd floor floorplan with sample trajectories.

The full-list of the testbeds' floorplans can be found here [75–79] and Table 4.1 summarizes the testbeds characteristics.

Building	# Floors	Area (ft^2)	# APs/Floor	# Samples
Centreville Hall - **98**	9	76,340	[8, 12, 10, 11, 11, 10, 9, 12, 17]	1837
Cumberland Hall - **122**	9	74,980	[11, 10, 15, 10, 12, 8, 18, 8, 15]	2280
Bel Air Hall - **99**	5	17,710	[5, 4, 6, 4, 8]	223
Cambridge Hall - **96**	5	34,631	[7, 8, 11, 8, 11]	1088
A.V. Williams - **115**	4	152,130	[36, 40, 38, 20]	4995
Total Number of Samples				10423

Table 4.1: Summary of the five testbeds and evaluation data.

For the rest of this chapter, we start by evaluating *Hapi*'s floor-level detection performance. Then, we evaluate its 2D location estimation accuracy. In addition, we compare its accuracy to two state-of-the-art calibration-free systems: Locus [57] and IncVoronoi [52].

4.2 Floor-Level Detection Accuracy

In this section, we evaluate the performance of *Hapi*'s floor-level detection performance. We start by evaluating the effect of the floor-level detection module's different parameters and components. Then, we show *Hapi*'s performance in the 5 testbeds as compared to Locus [57]. *Note that, we do not compare to IncVoronoi [52] here as it assumes a single-floor area of interest and estimates the user 2D location only.*

4.2.1 Effect of the Profile Window Size

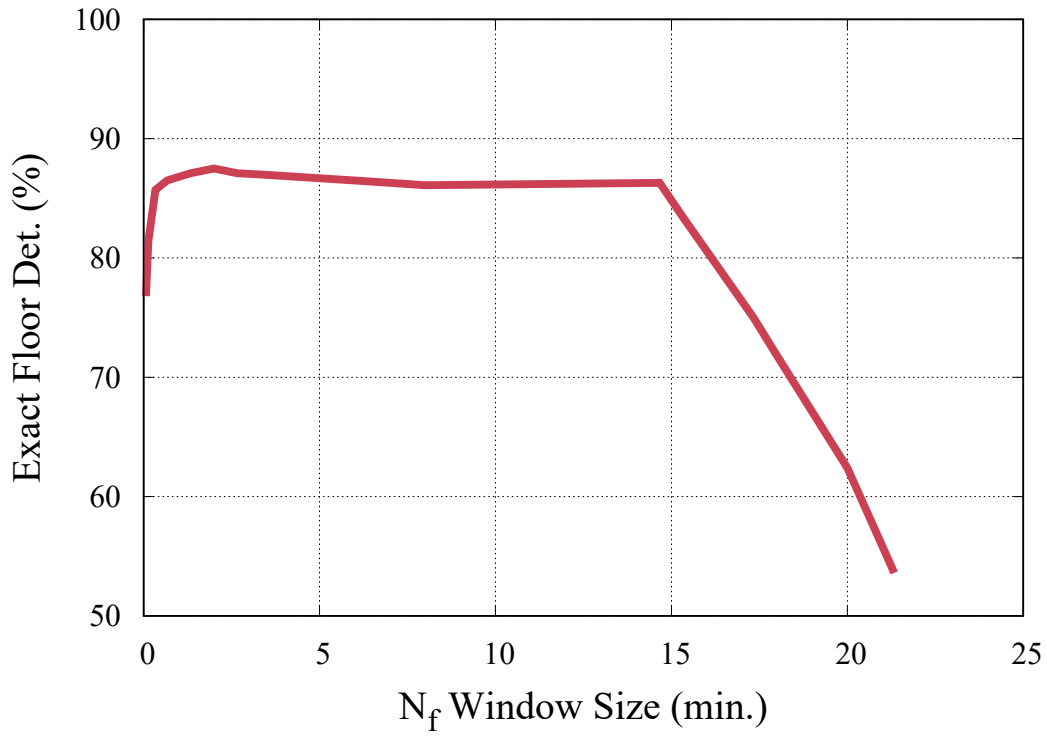


Figure 4.2: Effect of the Floor-level detection N_f value on *Hapi*'s exact floor detection accuracy.

Figure 4.2 shows the effect of the profile window size N_f on *Hapi*'s exact floor detection accuracy. *Hapi*'s WiFi profile helps better capture the user surrounding WiFi network. Hence, increasing N_f increases the system's exact floor-level detection accuracy. However, as the window gets too large, the accuracy starts decreasing; as the likelihood of the user moving to another floor increases with the length of the N_f window. We set N_f to 2 minutes as it balances accuracy and being a reasonable duration for the user dwelling period in a single floor.

4.2.2 Performance of the Different WiFi Features

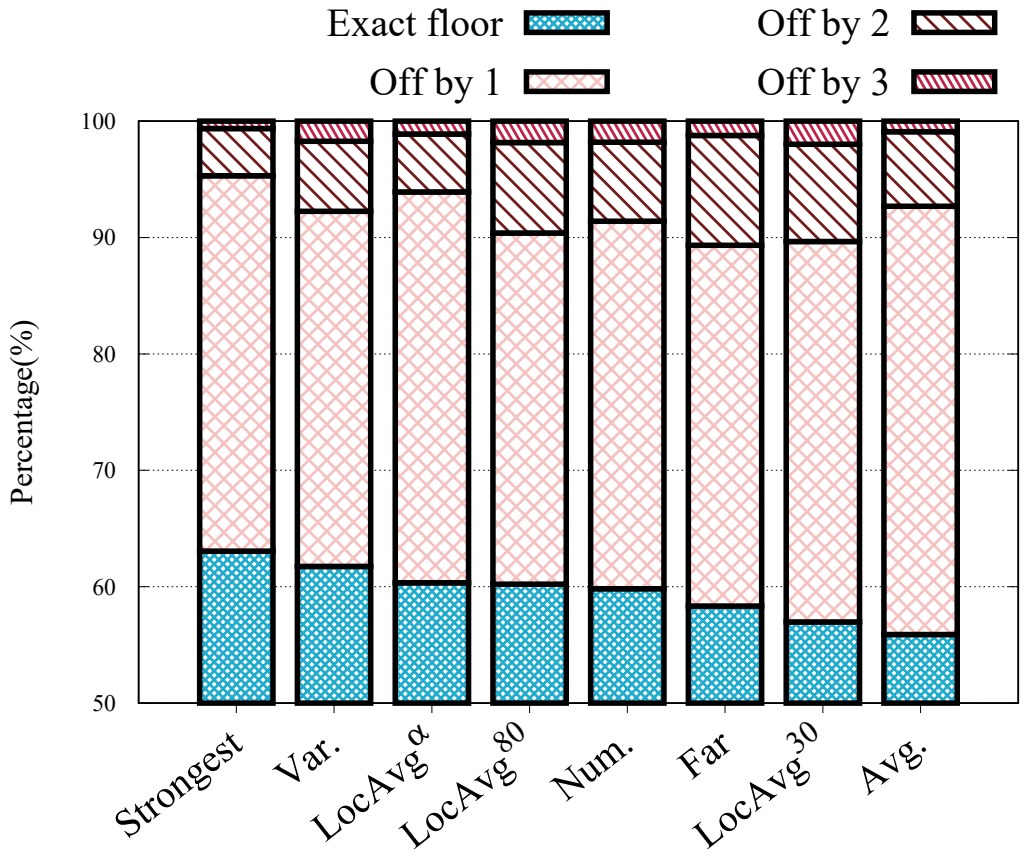


Figure 4.3: Performance of the extracted WiFi features overall the five testbeds.

Figure 4.3 shows the overall performance of the different WiFi features for the 5 testbeds. Using each feature, we set the user-floor as the floor with the maximum feature value. For example, for the "Number of APs" feature (*Num*), the user-floor becomes the floor with the maximum number of visible APs. Generally, we can identify the user floor around 60% of the time using any of the features. However, as you can see in Figure 4.5, each performed differently in the different testbeds due to their varied WiFi network and building characteristics. As we can see in the figure, using the APs' 2.5D location through the proposed *Local Average Signal Strength* (*LocAvg*) feature performed better than using the visible APs' average signal strength (*Avg*). In addition, selecting the proximity region using α performed better than other values because it varies based on the user profile.

4.2.3 Effect of our DL Training Methodology

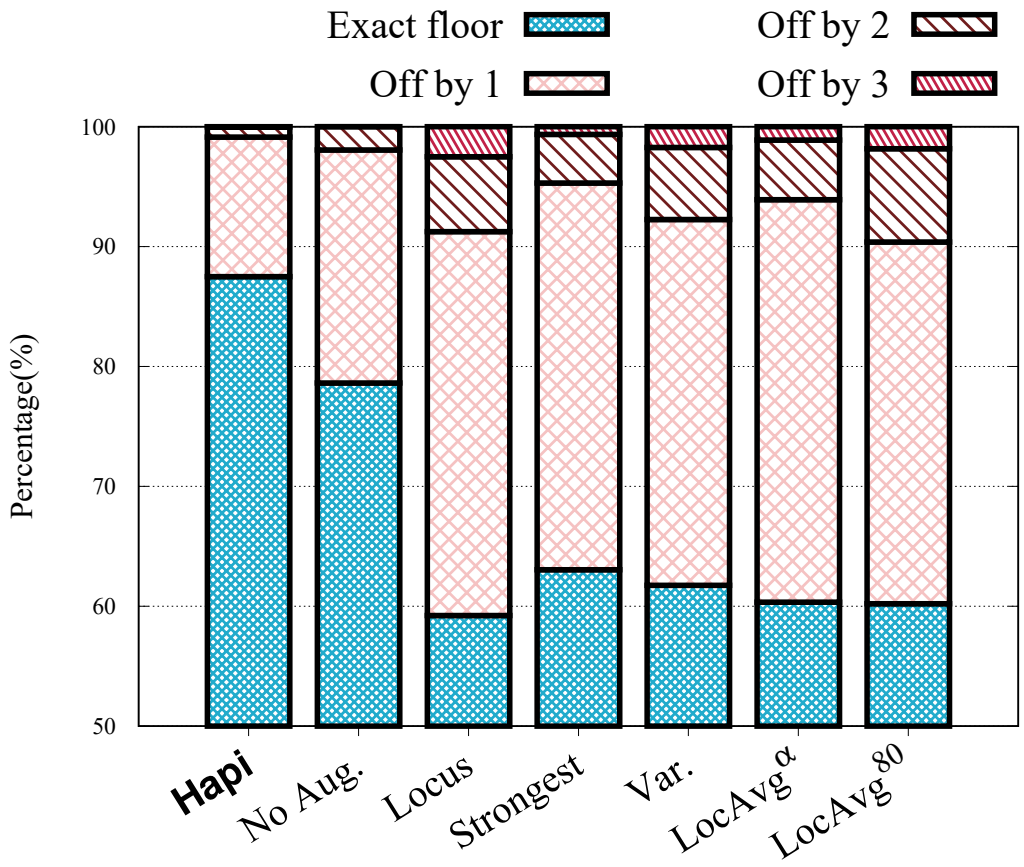
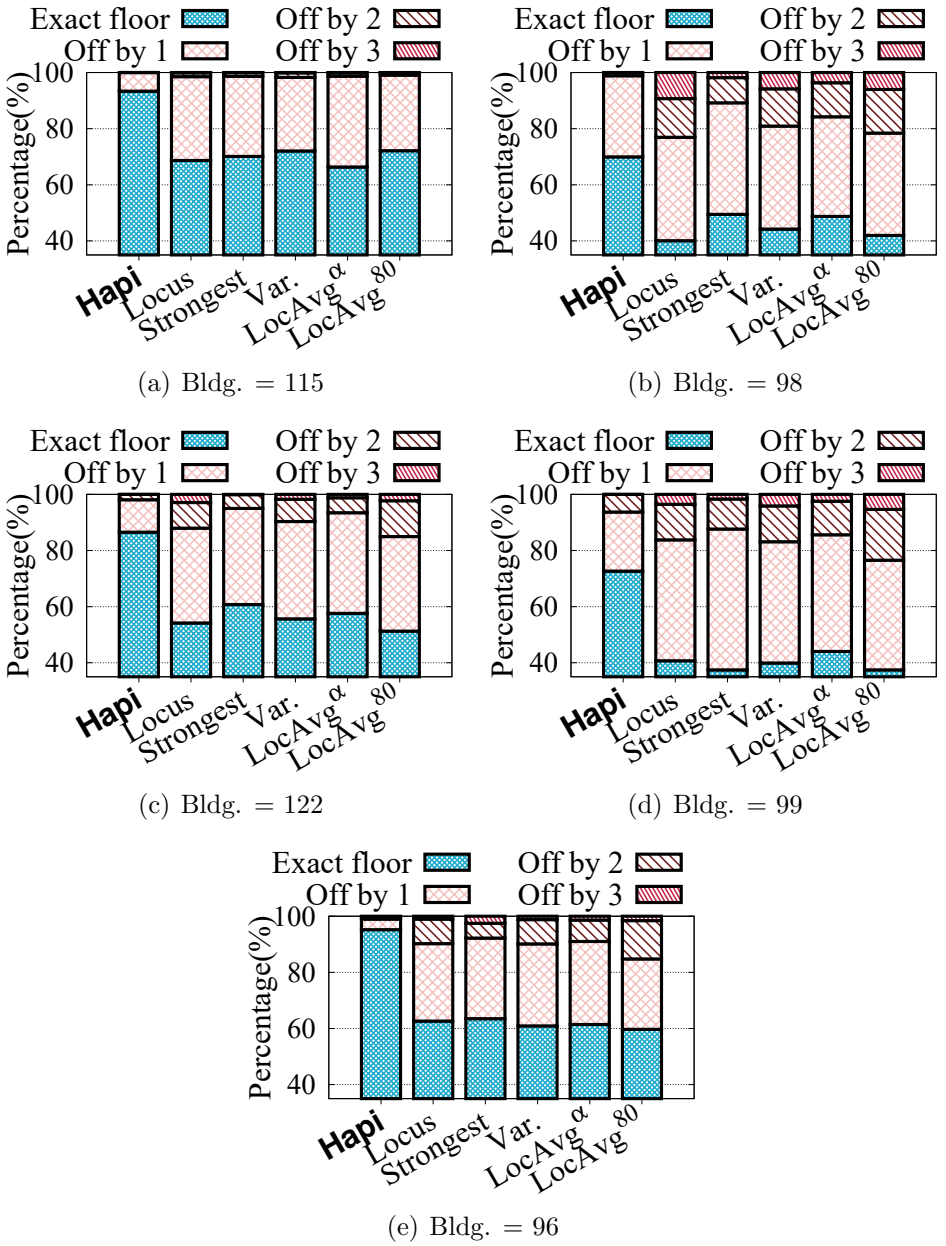


Figure 4.4: Effect of disabling training method on *Hapi*'s performance (No Aug. bar) as compared to Locus and the top performing four features.

Figure 4.4 compares *Hapi*'s floor-level detection to same DNN architecture without the data generalization and balancing during training (*No Aug.* bar on figure), Locus and the top performing four features. We can see that the DNN-method improves the accuracy over Locus and the individual techniques. Moreover, our proposed data generalization and balancing lead to an improvement of 11.3% in the user exact floor detection.

4.2.4 Comparison with Other Systems

Figure 4.5 below shows *Hapi*'s floor-level detection accuracy as compared to Locus [57] and top performing WiFi features in the 5 testbeds and Table 4.2 summarizes the results.



(a) Bldg. = 115

(b) Bldg. = 98

(c) Bldg. = 122

(d) Bldg. = 99

(e) Bldg. = 96

Figure 4.5: *Hapi*'s floor-level detection accuracy in the 5 testbeds as compared to Locus [57] and top performing 4 features.

Testbed	Exact Floor-Level Detection Percentage			2D Location Median Error		
	Hapi	IncVoronoi [52]	Locus [57]	*Hapi*	IncVoronoi [52]	Locus [57]
Bldg. 99	**72.6%**	N/A	40.7% *(+78.4%)*	**3.1m**	5.1m *(+39.2%)*	23.5m *(+86.8%)*
Bldg. 96	**95.2%**	N/A	62.6% *(+52.1%)*	**3.4m**	6.0m *(+43.3%)*	13.9m *(+75.5%)*
Bldg. 122	**86.5%**	N/A	54.2% *(+59.6%)*	**3.5m**	6.2m *(+43.5%)*	15.3m *(+77.1%)*
Bldg. 115	**93.4%**	N/A	68.7% *(+36.0%)*	**3.6m**	11.8m *(+69.5%)*	13.0m *(+72.3%)*
Bldg. 98	**70%**	N/A	40.1% *(+74.6%)*	**3.5m**	6.5m *(+46.2%)*	17.9m *(+80.4%)*
Overall	**87.5%**	N/A	59.2% *(+47.7%)*	**3.5m**	8.9m *(+60.7%)*	14.6m *(+76.0%)*

Table 4.2: Summary of *Hapi*'s accuracy as compared to IncVoronoi [52] and Locus [57]. Percentages are calculated relative to comparison systems.

We can see in the figure, *Hapi* beats the comparison systems in terms of exact floor detection accuracy as well as reducing maximum floor estimation errors. *Hapi*'s deep-learning model takes the building's network architecture into account and yields a better performance compared to individual features achieving up to 95.2% exact floor detection (in Bldg. 96) with no training data from any of the testing buildings. In addition, this is better than Locus [57] by 52.1%. Locus [57] algorithm was tuned using heuristics from experimentation in one building, this lead to decrease in its accuracy when applied in larger-scale. Also, Locus uses only the APs' installation floor, however, as shown in this chapter, using the APs' 2.D location can help improve the user floor level estimation.

4.3 2D Localization Accuracy

In this section, we evaluate *Hapi*'s 2D location estimation performance. We start by evaluating the effect of the 2D location estimation module's parameters and components. Then, we show *Hapi*'s performance in the 5 testbeds as compared to IncVoronoi [52] and Locus [57].

4.3.1 Effect of the Profile Window Size

Figure 4.6, below, shows the effect of the profile's N_l value on the user 2D location estimation performance.

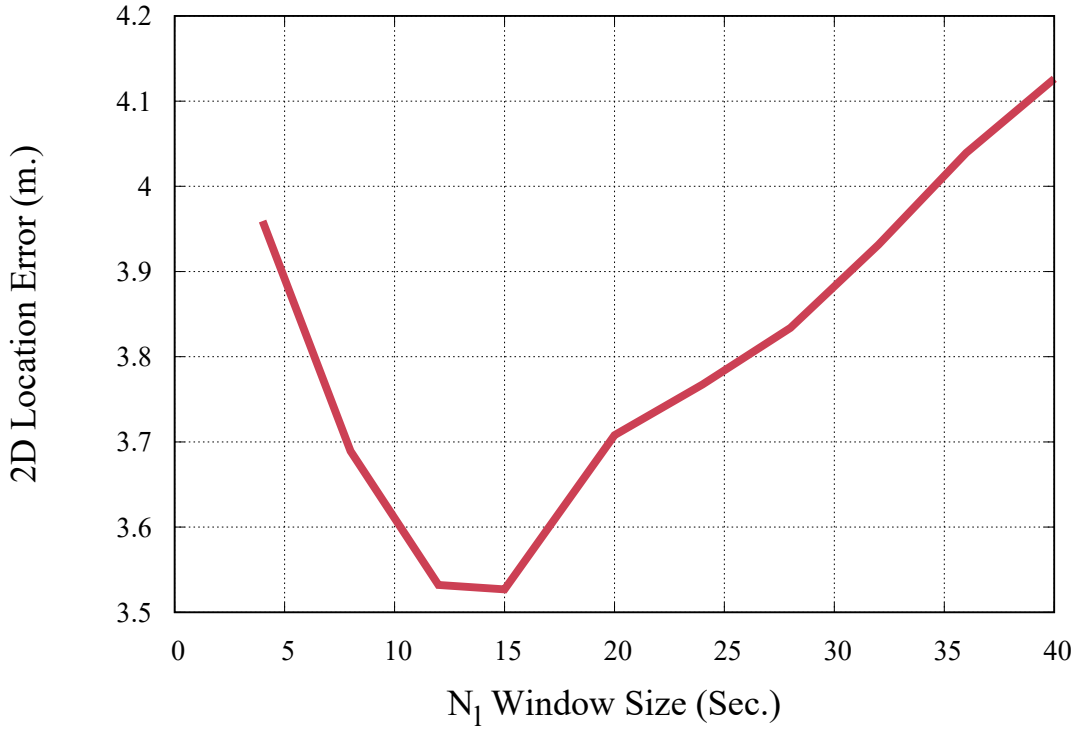


Figure 4.6: Effect of the N_l value on *Hapi*'s 2D location estimation accuracy.

The WiFi profile helps better capture the user surrounding WiFi network and thus a larger N_l leads to a better localization accuracy. However, as N_l increases, the user may have moved away from her location leading to accuracy reduction. Thus, we choose $N_l = 15$ seconds. Note that, as expected, the N_l value is shorter than the N_f value (Section 4.2.1), as users are more likely to stay longer on the same floor as opposed to their location within the floor.

4.3.2 Effect of the Floor Attenuation Factoring

Figure 4.7, below, shows the effect of the *Floor Attenuation Factoring* module on *Hapi*'s 2D location estimation accuracy.

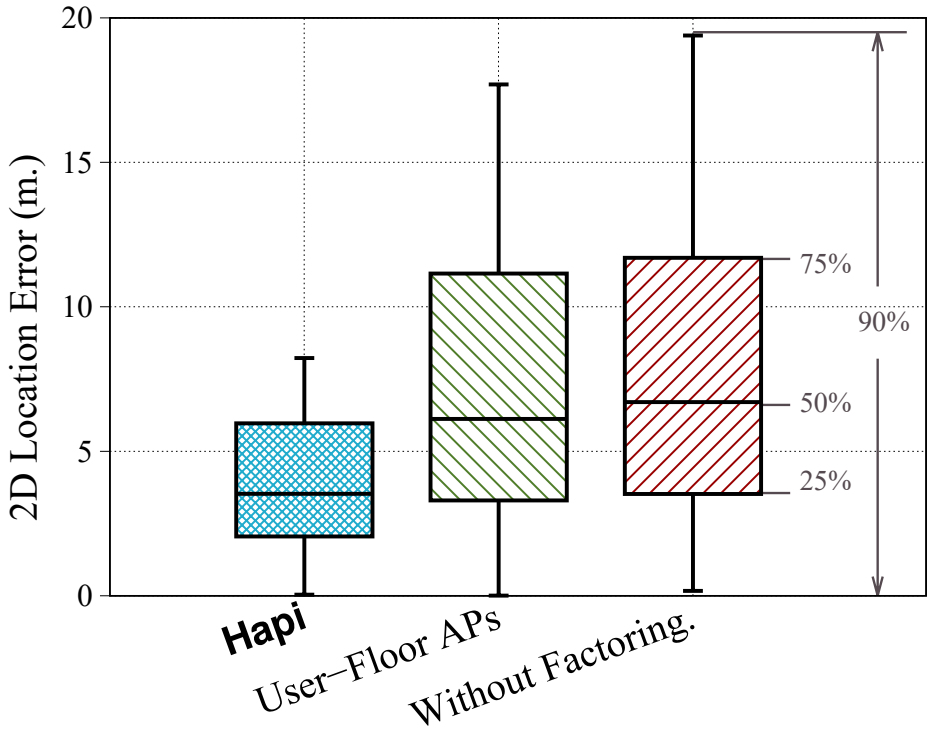


Figure 4.7: Effect of disabling *Hapi*'s Floor Attenuation Factoring module and using APs from the user floor only.

The figure emphasizes the importance of using visible APs from all floors while applying the Floor Attenuation Factoring. We can see that it improves the accuracy over just using APs from the user floor by 42.6% and over using the APs without the Floor Attenuation Factoring by 47.8%. Note that using APs from other floors without processing their RSS values leads to a decrease in the accuracy as they represent noisy/misleading measurements added to the user location PDF as

discussed in Section 3.4.1.

We have also analyzed the effect of the Floor Attenuation Factoring module W_f parameter on *Hapi*'s 2D location estimation accuracy as shown in Figure 4.8 below.

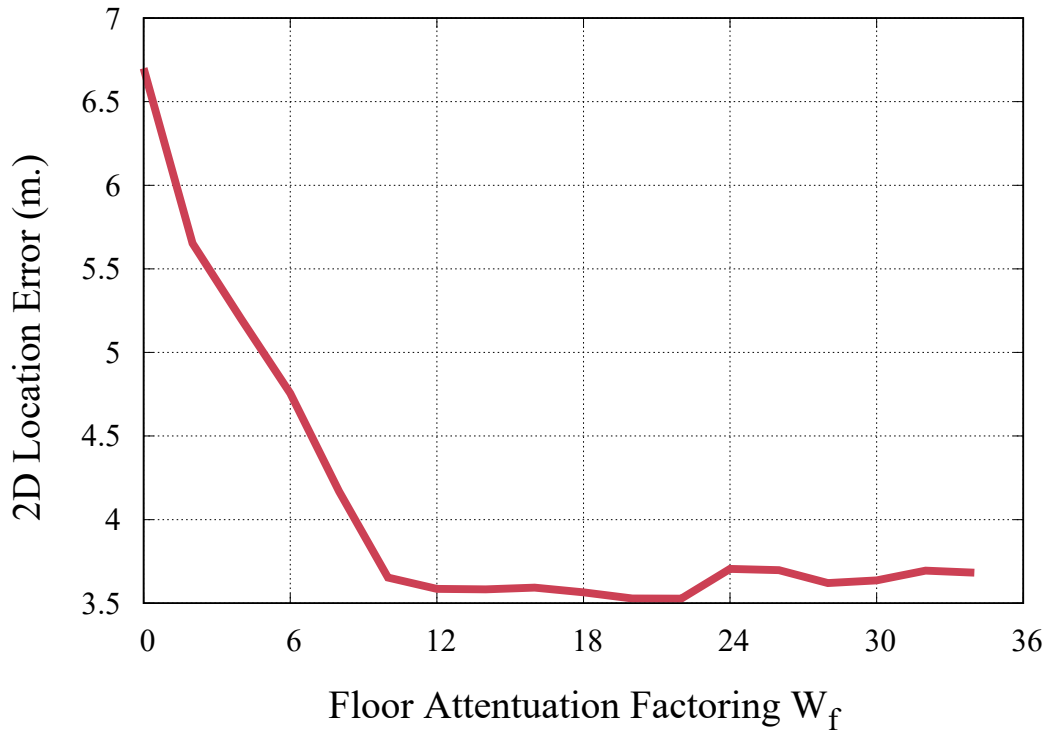


Figure 4.8: Effect of the W_f value on *Hapi*'s 2D location estimation accuracy.

Increasing W_f improves the accuracy as it compensates for the AP's RSS attenuation due to its installation floor. Yet, as it increases, it can end up mapping all RSS values from APs in other floors to a Very Strong Rank which causes a decrease in accuracy. We set W_f to 15 to balance the RSS mapping as seen in the figure.

4.3.3 Effect of the Location Estimation Threshold

Figure 4.9 shows the effect of the location estimation threshold (ℓ_{thr}) on *Hapi*'s location estimation accuracy.

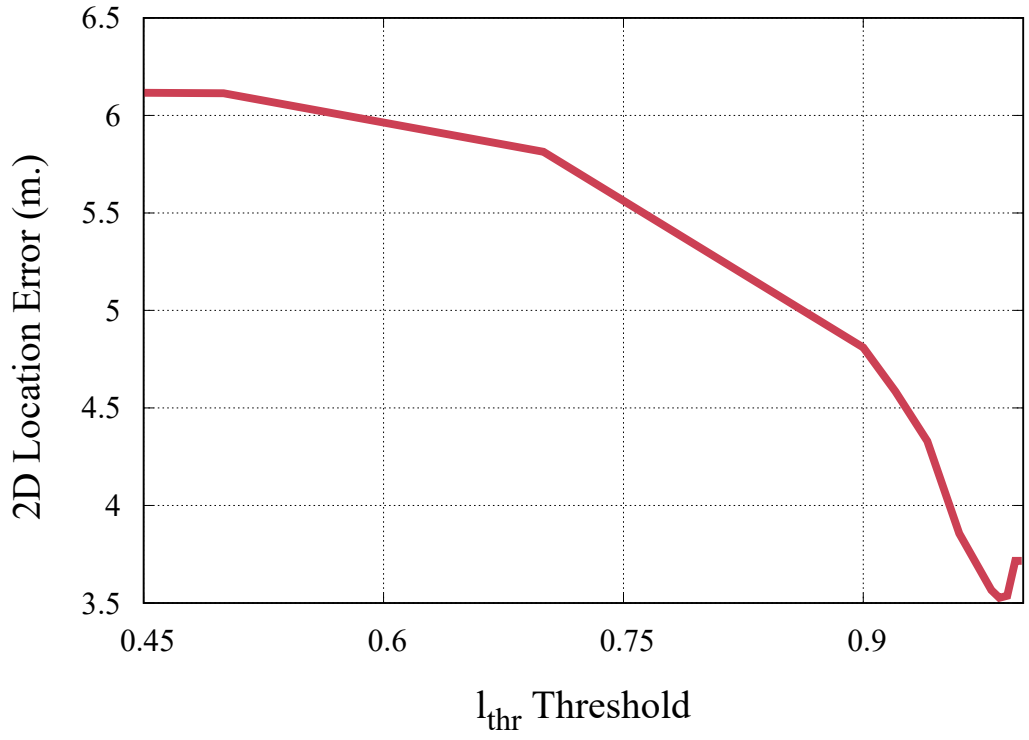


Figure 4.9: Effect of the location estimation ℓ_{thr} on *Hapi*'s 2D location estimation accuracy.

Increasing ℓ_{thr} leads to removing uncertain areas (i.e. areas where the user is most probably not located) when calculating the user location and consequently improving the system's location estimation accuracy. However, as ℓ_{thr} increases, it may lead to removing areas where the user can be located and decreasing the localization accuracy. Thus, we set ℓ_{thr} to 0.98.

4.3.4 Effect of the KF Refinement

Figure 4.10 shows the effect of the KF Location Refinement on *Hapi*'s 2D location estimation accuracy.

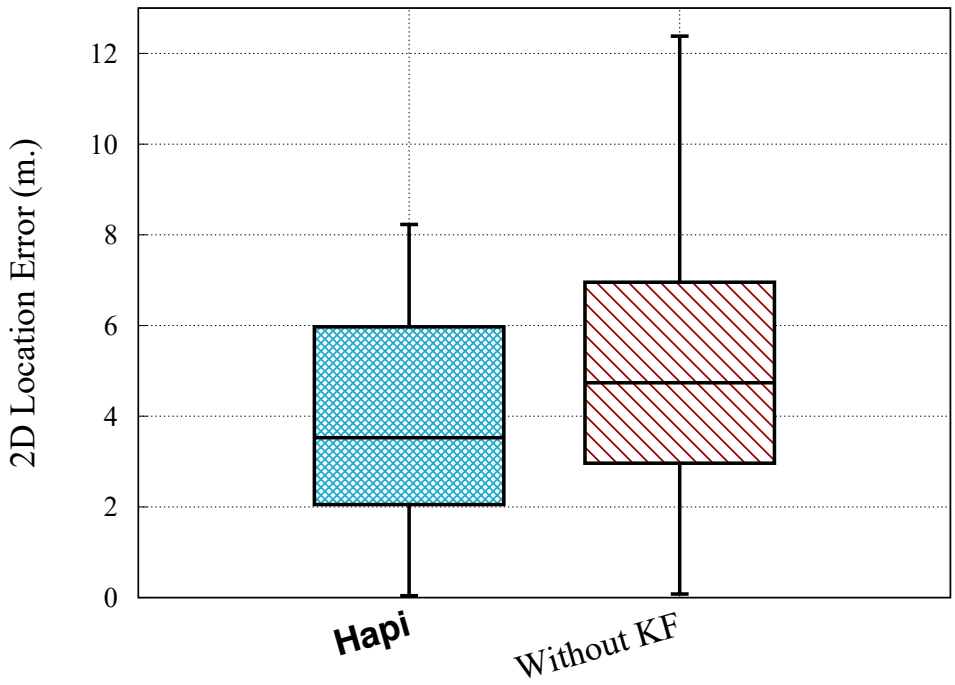
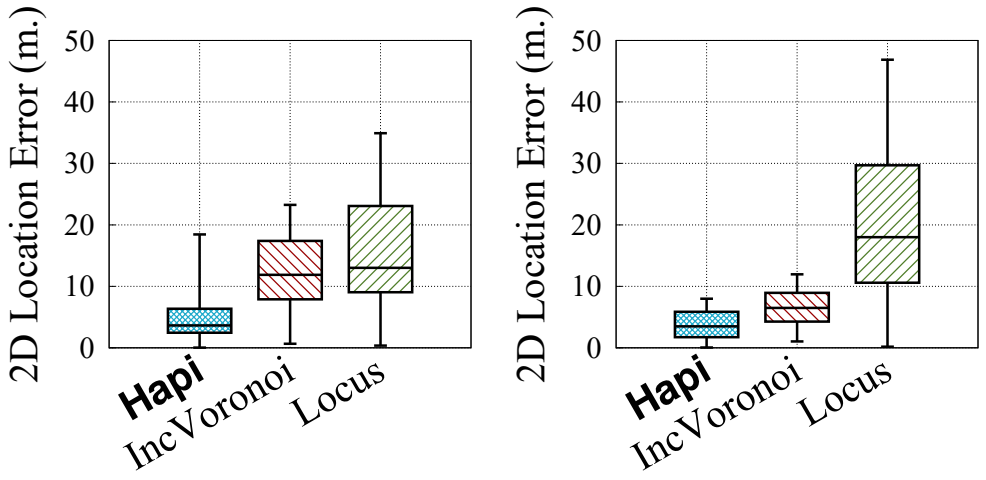


Figure 4.10: Effect of the Kalman Filter module on *Hapi*'s 2D location estimation accuracy.

The KF reduces the maximum errors significantly and the median error by 25.5%, as it considers both the user's motion history and the quality of the location instances.

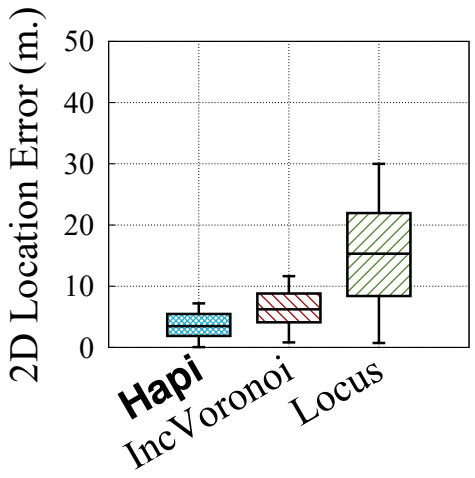
4.3.5 Comparison with Other Systems

Figure 4.11 shows *Hapi*'s 2D localization accuracy in the 5 testbeds and compares it to IncVoronoi [52] and Locus [57]. Table 4.2 summarizes the results.

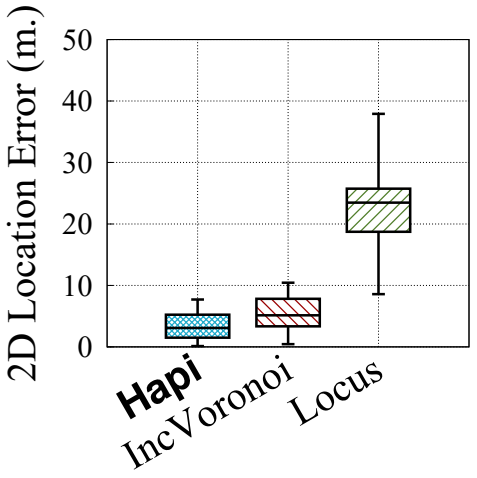


(a) Bldg. 115

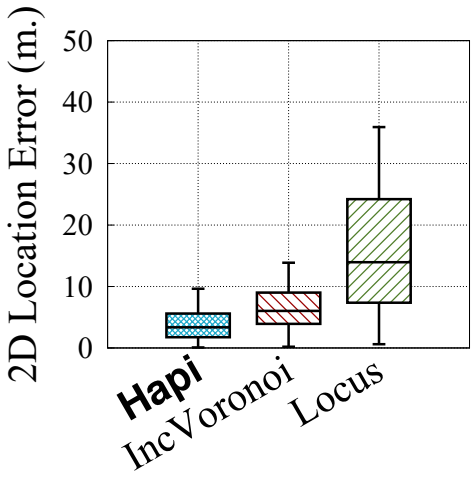
(b) Bldg. 98



(c) Bldg. 122



(d) Bldg. 99



(e) Bldg. 96

Figure 4.11: *Hapi*'s 2D localization accuracy in the 5 testbeds as compared to IncVoronoi [52] and Locus [57].

As shown in the figure above, *Hapi* provided a better accuracy in all testbeds over the comparison systems. We can see that Locus using, only, the user-floor's APs' raw RSS values through triangulation resulted in a lower accuracy when compared to *Hapi* and IncVoronoi [52] relative RSS approaches. Specifically, *Hapi* had 76.0% improvement in median accuracy over Locus overall testbeds. While IncVoronoi had better performance than Locus, using the user floor to preprocess the RSS and our proposed novel RSS-Rank Gaussian model enabled *Hapi* to improve the median accuracy consistently and achieving an improvement of 60.7% overall testbeds.

4.4 Summary

In this chapter, we discussed the implementation of the *Hapi* system and our experimental evaluation results as well as a comparison with state of the art smartphone-based calibration-free WiFi indoor localization systems (IncVoronoi [52] and Locus [57]). We have also analyzed the *Hapi* system parameters and reasoned the parameters selected values, which we used throughout the evaluation. *Hapi* was evaluated in five different multistory buildings with up to nine floor tall buildings. Thirteen subjects collected data over a period of six months using a wide range of smartphone devices.

We evaluated *Hapi*'s floor-level detection accuracy as compared to Locus [57] (as IncVoronoi [52] works on a single floor only). *Hapi* achieved up to 95.2% exact floor detection accuracy in a calibration-free deployment (i.e. without requiring

any training data from the deployment building). In addition, this is better than Locus [57] by 52.1%. We have also shown that our proposed neural network method leads to an improvement of 11.3% in the user exact floor detection performance.

We also evaluated *Hapi*'s 2D location estimation accuracy as compared to IncVoronoi [52] and Locus [57]. *Hapi* had a median localization error of 3.5m this is better than IncVoronoi [52] by 60.7% and better than Locus [57] by 76.0%.

Chapter 5: Accessibility Semantics Detection

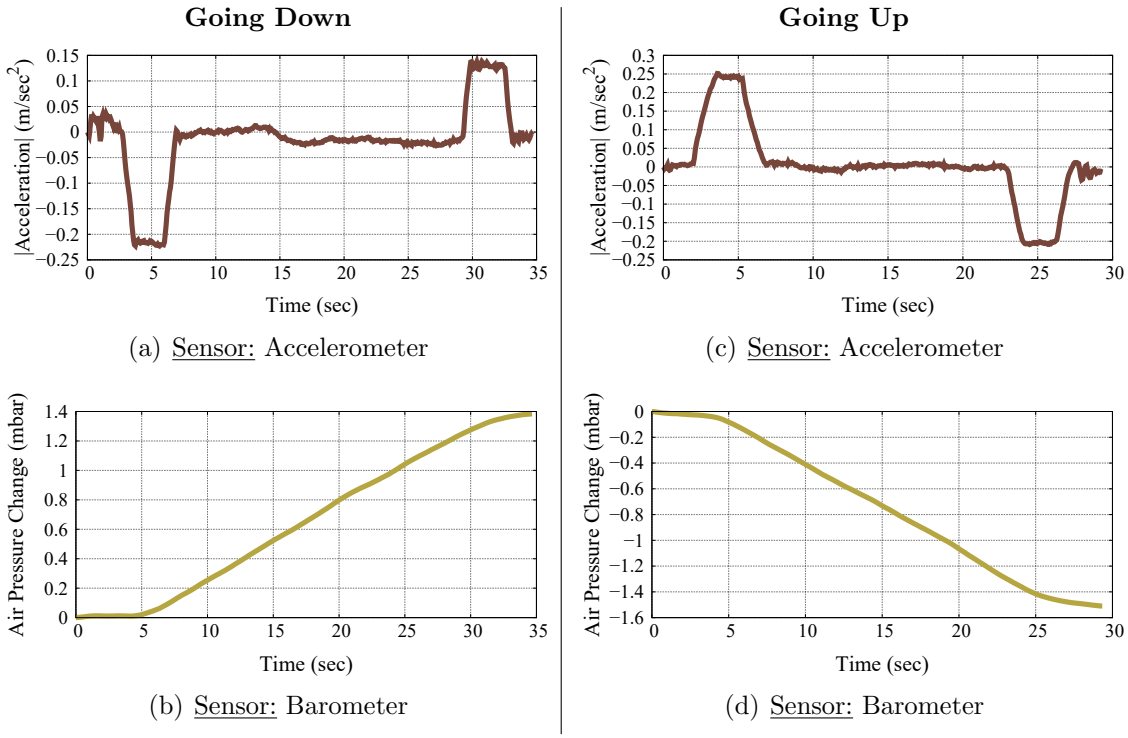
5.1 Overview

AccessMap employs a modular architecture for the accessibility semantics detection module. In this chapter, we present the algorithms used by *AccessMap* to detect the different accessibility semantics via passive crowd-sourcing.Specifically, we present algorithms to detect elevators, visually-impaired accessible elevators, accessible pedestrian signals, curbs, staircases, and ramps. Additionally, we discuss how those detected semantics affect the space's accessibility assessment.

5.2 Elevators

Elevators are necessary for wheel-chaired individuals to move between floors. Thus, *AccessMap* detects elevators to rate access to floors above and below the ground, such as underground metro-station entrances and floors in multistory buildings, as **wheel-chaired accessible**. When a person takes an elevator, it affects multiple sensors in her mobile device. Specifically, as the elevator moves rapidly between floors, it induces a rapid change in the air-pressure captured by the barometer (Figures 5.1(b) and 5.1(d)). Additionally, the accelerometer gets affected by

the elevator movement; leading to a peak or valley in the acceleration magnitude based on the elevator's moving direction. This is followed by a no acceleration (due to standing in the elevator) then the peak or valley reverses as the elevator stops (Figures 5.1(a) and 5.1(c)).



(a) Sensor: Accelerometer

(c) Sensor: Accelerometer

(b) Sensor: Barometer

(d) Sensor: Barometer

Figure 5.1: Effect of using the elevator on the acceleration magnitude and the air pressure change while going up and down.

5.2.1 Detection Algorithm

To detect elevators from the crowd-sourced traces, we start by segmenting the traces. The typical approach of a fixed window size is inadequate to capture the acceleration pattern shown in Figures 5.1(a) and 5.1(c). Thus, we dynamically segment the traces based on the barometer readings to detect periods where users are moving between floors. For that, we compute the slope of the barometer's air-

pressure and identify segments with high slope as candidate elevator segments. To put it formally, for a crowdsourced trace $C = \{c_1, c_2, c_3, ...\}$: we extract a segment $s = [c_a : c_e]$ of variable length $L = e - a + 1$ such that $\forall_{c \in s}\ \delta c.p \geq \alpha_p$, $\delta c_{a-1}.p < \alpha_p$ and $\delta c_{e+1}.p < \alpha_p$ where $c_t.p$ is the barometer sensor measurement at t and α_p is a system parameter for the air-pressure slope. Then, a finite state machine is employed to detect the elevator pattern (Figure 5.1) from the acceleration magnitude

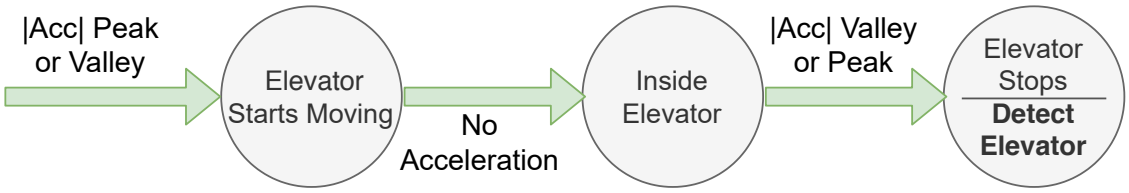


Figure 5.2: A finite state machine to detect the elevator acceleration magnitude pattern.

5.3 Visually-impaired Accessible Elevators

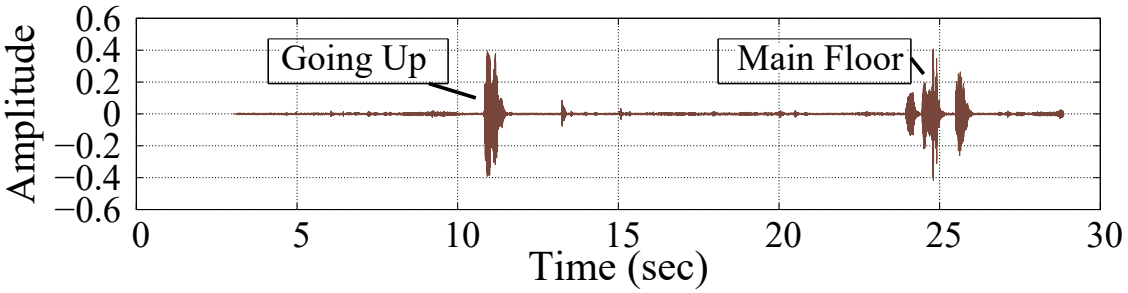
While elevators help wheel-chaired individuals move vertically between floors (as discussed in the previous section), riding them can be confusing/challenging for visually-impaired individuals. This is due to two reasons: First, elevators display their moving direction through a visual up/down arrow (e.g. Figure 5.3(a)) ; making it hard for a visually-impaired individual to decide whether to take an elevator or wait for the next one. Second, while elevators typically have Braille signage, to help requesting a floor, the current floor is displayed throw a visual seven segment display (SSD) board (e.g. Figure 5.3(b)) which again makes it hard for a visually-impaired individual to know whether they should get off the elevator or wait.

(a) Moving Direction Hints

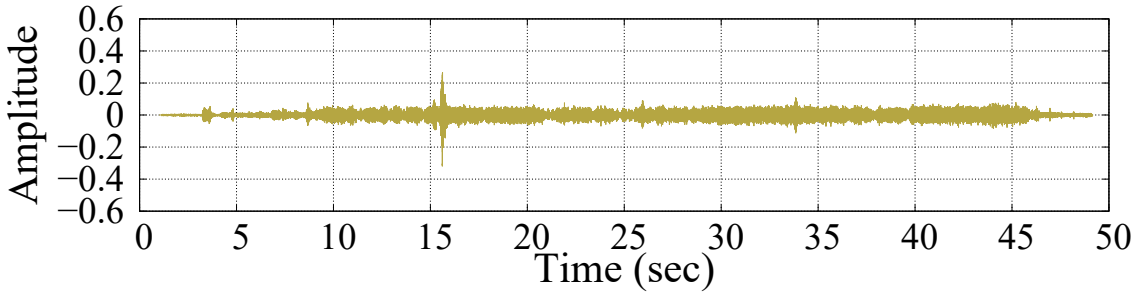
(b) Current Floor Hints

Figure 5.3: Elevators depend on visual hints to indicate their moving direction (E.g. arrow pointing up or down) and the current floor (E.g. 7-Segment indicator displaying the floor number).

To address these issues, recently, visually-impaired accessible elevators are deployed in buildings. These elevators communicate their moving direction and current floor through audio announcements such as "Going Up" and "Third floor" respectively. Thus, *AccessMap* detects the announcements to rate elevators as **visually-impaired accessible**. To capture those announcements, *AccessMap* relies on the phone's microphone sensor. For example, as shown in Figure 5.4(a), audio captured while taking a visually-impaired accessible elevator shows a "Going up" announcement as the user walks into the elevator and a "Main floor" announcement as the elevator reaches its floor and stops. On the other hand, for inaccessible ones, no audio announcements were captured as shown in Figure 5.4(b).



(a) Visually-impaired **accessible**



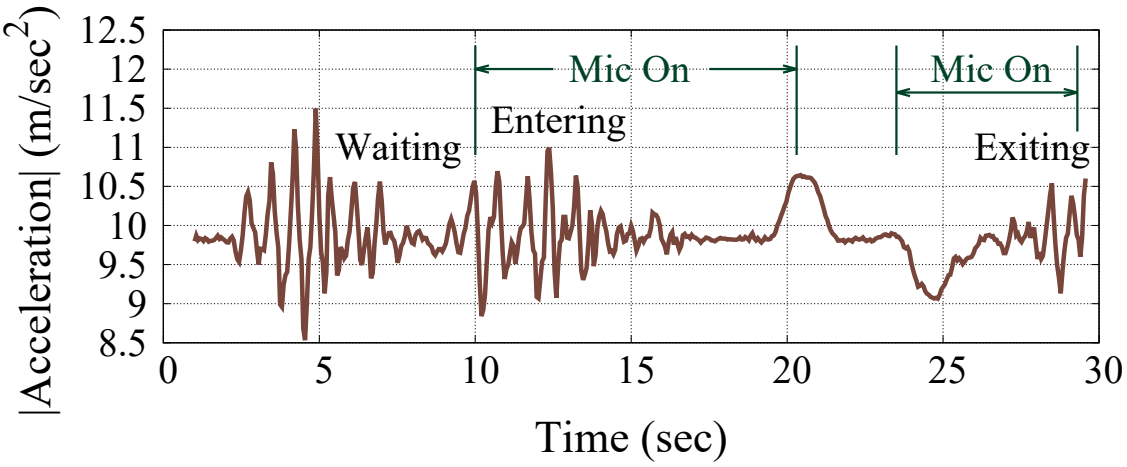
(b) Visually-impaired **inaccessible**

Figure 5.4: Audio captured while taking a visually-impaired accessible elevator vs inaccessible one. The accessible elevator announces its moving direction ("Going up") and its current floor ("Main floor").

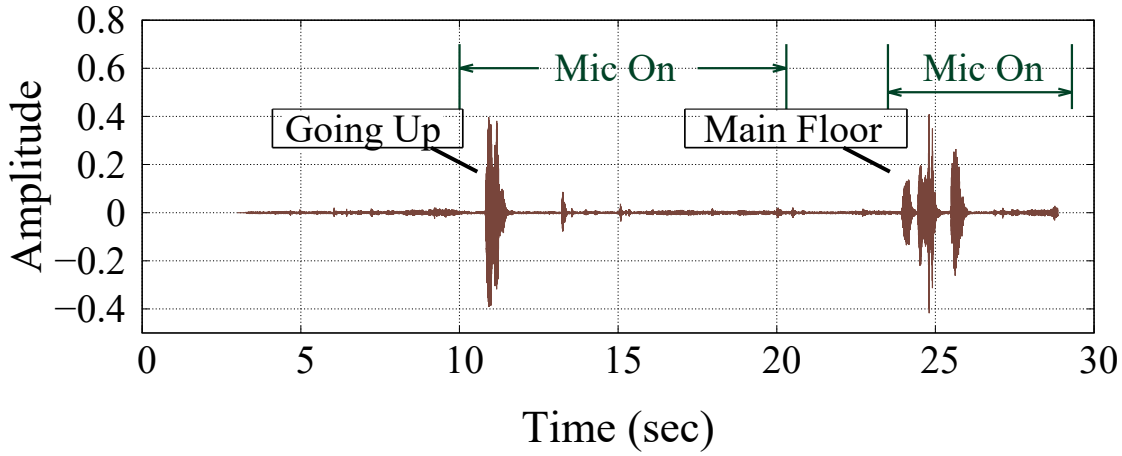
5.3.1 Detection Algorithm

To detect the audio announcements, we rely on Automatic Speech Recognition (ASR) to recognize whether a moving direction is announced (i.e. "Up" or "Down") before the elevator starts moving and a stopping floor (e.g. "Floor" or "Level") as it stops. We employed a pre-built deep-learning neural network for video transcription from Google's speech to text models [80, 81].

To collect the audio-hints, while minimizing the time the microphone is on, we rely on the mobile device's inertial sensors to turn-on/off the microphone (Figure 5.5).



(a) Acceleration signal



(b) Elevator audio hints

Figure 5.5: Acceleration events while taking the elevator can help minimize the time the microphone is on.

Specifically, we detect the dwelling nearby the elevator (i.e. waiting for the elevator) and start the microphone as the user walks towards the elevator till the elevator starts moving (peak or valley in the acceleration magnitude) and then start the microphone again after the elevator stops (reversed peak or valley in the acceleration magnitude) till the user walks out of the elevator.

5.4 Accessible Pedestrian Signals

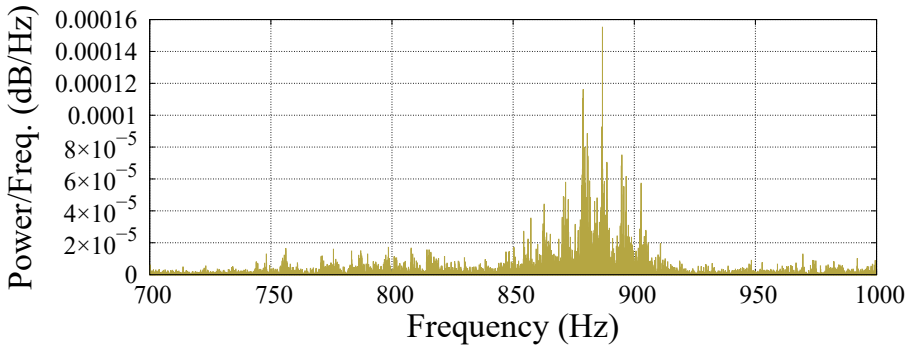
Accessible pedestrian signals are integrated devices that communicate the stop-and-go information at signalized crosswalks and intersections in non-visual formats. This makes the signals accessible for pedestrians who are blind or suffer from low vision. In general, such signals let pedestrians know: (1) they can cross the road through an audible clear cue that starts at the beginning of the pedestrian phase, and (2) they should wait through a different audible cue (it can start when the signal's button is pushed). Both cues can be tones or verbal messages which have standardized formats through the rights-of-way accessibility guidelines [82].

For audio tones, they can consist of multiple frequencies, high and low, with a dominant component at 880 Hz. As this frequency has been found to be highly detectable and localizable in the presence of traffic sound. Additionally, frequencies above 1kHz are difficult to detect for persons with age related upper-frequency hearing loss. On the other hand, the speech messages are required to follow a recommended model. For example, for pedestrian phase announcement, it should include "WALK sign is on" and for the vehicular phase, it should include "Wait". Note that, the audio clue's volume is responsive to ambient noise level changes; i.e. it can get louder based on the ambient sound. Thus, *AccessMap* detects the audio-clues to rate pedestrian signals as **visually-impaired accessible**. When a pedestrian uses an accessible signal, her mobile device's accelerometer and microphone captures the stop-and-go walking pattern and the audio clues respectively as shown in Figure 5.7. So, both sensors are utilized by *AccessMap* to detect the clues.

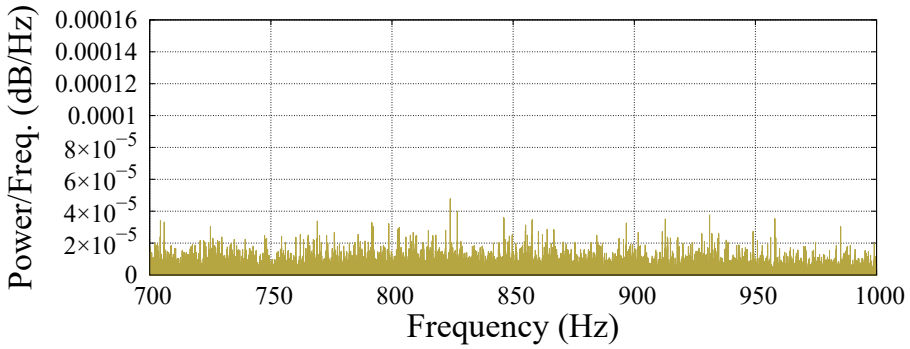
5.4.1 Detection Algorithm

AccessMap analyzes the audio recordings to search for either speech messages or audio tones. First, to detect the speech messages, we rely on Automatic Speech Recognition (ASR) to recognize whether "Walk sign" or "Wait" is announced. Specifically, we use the same pre-built deep-learning neural network used for detecting the elevators announcements (Section 5.3).

On the other hand, to detect the audio tones, as discussed earlier, an accessible pedestrian signal produces multiple frequencies with dominant component at 880 Hz as shown in the periodograms in Figure 5.6.



(a) **Accessible** signal

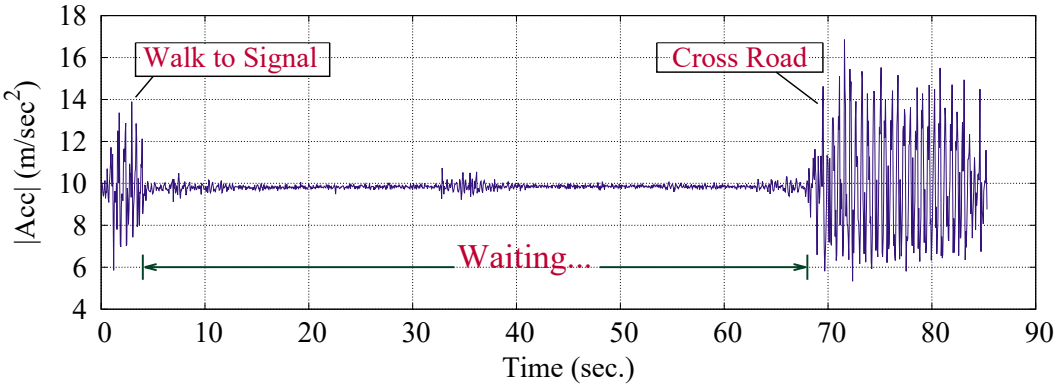


(b) **Non-accessible** signal

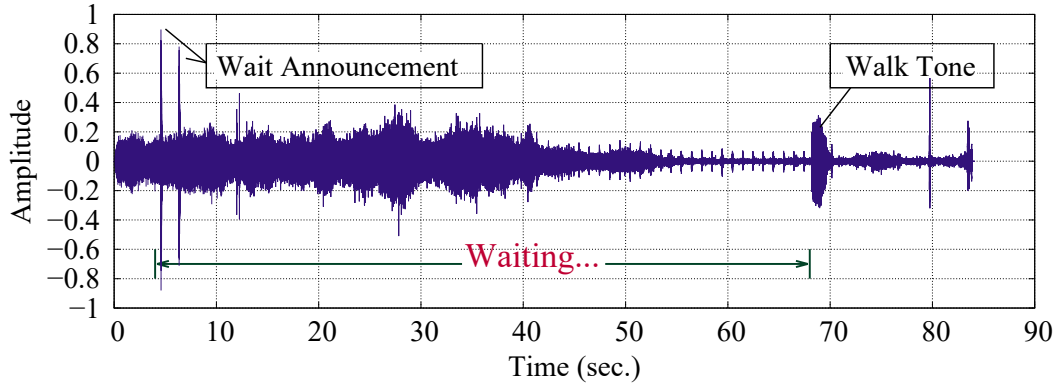
Figure 5.6: Example for accessible and non-accessible pedestrian signals periodograms around the 880 Hz frequency.

Thus, we start by transforming the audio signal s to the frequency domain (s_f) by applying Fast Fourier Transform (i.e. $s_f = FFT(s)$). Then, we apply a band-pass filter to attenuate traffic-noise frequencies and focus on the audio tone with a band from 700 Hz to 1000 Hz. Finally, to detect whether the frequency band (s_{f880}) contains a tone or not, we extract the frequency of the highest power ($\arg\max(s_{f880})$) and the ratio of its power to the mean power($\frac{max(s_{f880})}{mean(s_{f880})}$). The features are fed to a decision tree [83] to detect the tone.

To minimize the time the microphone is turned on, *AccessMap* utilizes the acceleromter (Figure 5.7).



(a) Acceleration signal



(b) Audio cues signal

Figure 5.7: The phone's accelerometer captures the stop-and-go pattern, and its mic captures the audio cues from the signal.

As the user reaches the pedestrian signal, the microphone is turned on for T_s seconds to capture any announcements for the vehicular phase. Then, to capture the phase announcements, the microphone is turned on again for another T_s seconds when the user starts walking. T_s is a system parameter, we empirically choose 5 seconds as it gave high recall as shown in Chapter 7.

5.5 Curbs

Curbs represent an obstacle for wheel-chaired individuals while stepping up/down from a sidewalk. Thus, curb-ramps are deployed on the top surface of sidewalks and graded down to the surface of their adjoining streets. Figure 5.8 shows example for a crosswalk with curb-ramps installed on both sides and another one without it. As seen in the figure, the deployed curb-ramps help make the sidewalk accessible for the wheel-chaired, individuals with walkers, etc.

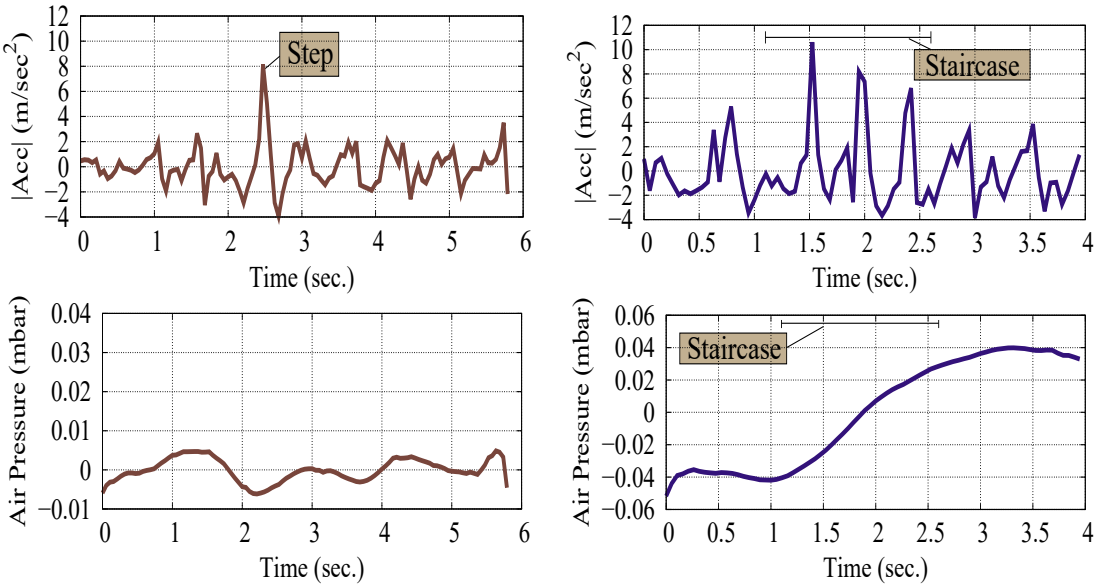
(a) Wheel-chaired **inaccessible** (Curb without ramp).

(b) Wheel-chaired **accessible** (Curb-ramps deployed on road sides).

Figure 5.8: Example for an inaccessible route vs. an accessible one for wheel-chaired individuals.

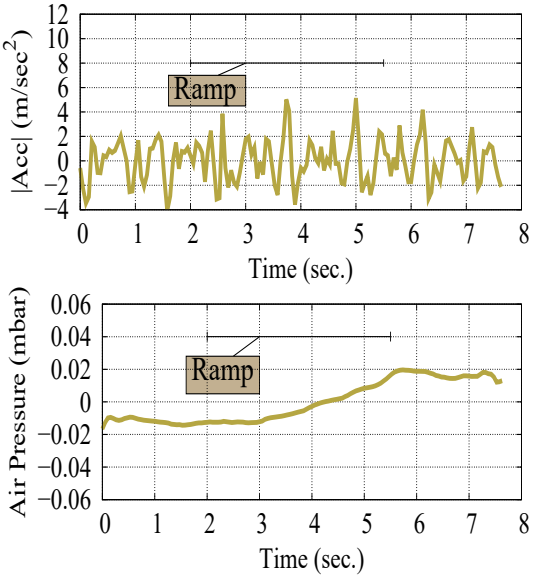
If a sidewalk's curb has a barrier (i.e. missing the curb-ramp) at intersections

or crosswalks, *AccessMap* assesses this part of the sidewalk as **inaccessible for wheel-chaired individuals**.



(a) Curb

(b) Staircase



(c) Ramp

Figure 5.9: Effect of a sidewalk curb, a staircase and a ramp on the mobile device's acceleration magnitude (|Acc|) and the barometer's air pressure sensor measurements. Note that, the values are normalized by subtracting the mean for comparison clarity.

Our analysis of the smartphone devices' sensors shows that as an individual steps up/down from a curb's barrier, her excreted force leads to a higher acceleration magnitude compared to walking on the crosswalk/sidewalk as shown in Figure 5.9(a). Yet, due to the minimal elevation change, the air pressure change, captured by the barometer, is also minimal compared to walking on other elevated semantics such as staircases and ramps (Figure 5.9). Therefore, *AccessMap* utilizes both barometer and accelerometer sensors to identify curbs.

5.5.1 Detection Algorithm

AccessMap levarges the observations discussed earlier to detect the sidewalk curb from the crowdsourced traces as follows: First, we start by segmenting the traces to steps segments using a step detection algorithm [72]. Then, we extract the variance of the acceleration magnitude and the air-pressure slope during the step segment. The features are fed to an SVM classifier which identifies whether the user was stepping on/down from a sidewalk or not.

5.6 Staircases

Stairs on a route such as a walkway or a doorway can be an obstacle to a wheel-chaired individual's independence, safety and quality of life. Thus, *AccessMap* **detects staircases and marks parts of routes that contain them as inaccessible for wheel-chaired individuals**.

When ascending or descending stairs, the gravity force affecting the person is

captured by her smartphone device's accelerometer sensor. Specifically, the gravity force leads to a higher peak in the acceleration signal, and hence higher variance, as compared to walking on a flat surface as shown in Figure 5.9(b). Additionally, stairs, typically, have high slope. Thus, asscending or descending a staircase leads to a decrease or increase in the air-pressure (based on the moving direction), that can be captured by the smartphone device barometer sensor. So, both the barometer and the accelerometer sensors are utilized to identify staircases.

5.6.1 Detection Algorithm

To detect stairs, again, *AccessMap* segments the crowdsourced traces to steps segments using a step detection algorithm [72]. Then, we extract the variance of the acceleration magnitude and the air-pressure slope. Note that, using the acceleration magnitude helps the detection algorithm be invariant to the phone attitude and position. The features are fed to an SVM classifier which detects whether there is a staircase or not.

5.7 Ramps

Ramps provide safe and reliable access to elevated surfaces for the wheel-chaired, individuals using power-chairs, people pushing strollers/carts, etc. Ramps are typically installed along with or instead of a staircase. Thus, *AccessMap* **detects ramps and marks parts of routes that contain them as accessible for wheel-chaired individuals**. When a person walks down a ramp, her mobile

device's barometer captures the elevation change (Figure 5.9(c)). Additionally, as seen in the figure, due to the typical ramp smoothness, the acceleration force excreted while walking is minimal. Hence, *AccessMap* employs both barometer and accelerometer sensors to identify ramps.

5.7.1 Detection Algorithm

To detect ramps, *AccessMap* uses the steps segments (extracted using the step detection algorithm [72]). Note that, different from staircases, ramps should be as shallow as possible. Thus, to characterize it, we measure the average air pressure slope in the segment and its variance as features. We also extract the variance of the acceleration magnitude. Then, these features are fed to an SVM classifier which detects whether there is a ramp or not in a given segment.

5.8 Summary

AccessMap employs a modular architecture for the accessibility detection module. Thus, different components can be added to detect various accessibility semantics and help increase the map's accessibility coverage. Moreover, based on the accessibility semantics characteristics, different sensors can be used. Also, based on the difficulty of the semantics detection, different levels of machine-learning techniques can be employed.

In this chapter, we provided the details for detection a variety of accessibility semantics: elevators, visually-impaired accessible elevators, accessible pedestrian

signals, curbs, staircases, and ramps. We also discussed how those semantics are used to assess the accessibility of their deployment spaces for wheel-chaired and visually-impaired individuals.

Chapter 6: Accessibility Map Construction

6.1 Overview

AccessMap constructs the accessibility map using a crowdsourcing approach incrementally, i.e. it keeps updating the map as it gets more crowdsourced data (Figure 6.1). For each newly observed semantic (detected by the Accessibility Semantics Detection module), *AccessMap* assesses whether it has been previously encountered, i.e. the newly detected semantic already exists in the map, through the *Accessibility Map Association* Module. Thereafter, the map is updated, corresponding to the semantic's association, using the *Accessibility Map Update* Module.

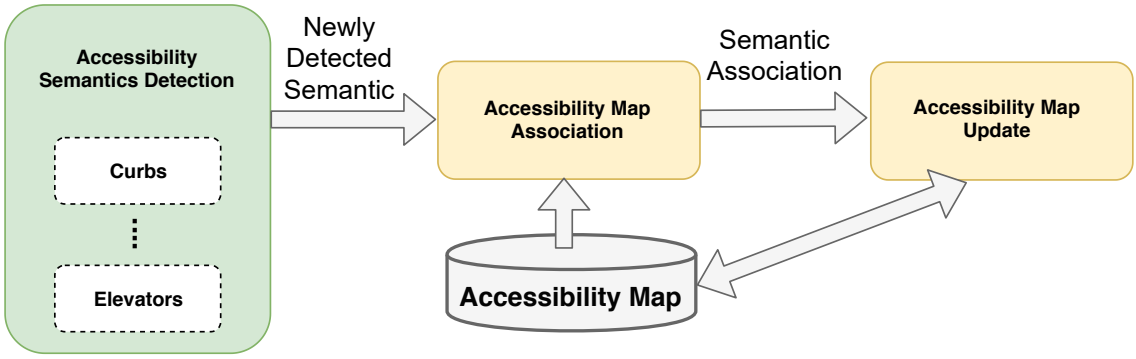


Figure 6.1: Overview of *AccessMap*'s accessibility map construction approach.

We describe the incremental map construction algorithm details in this chapter and start with the map's mathematical model.

6.2 Accessibility Map Model

AccessMap models the accessibility map (A) as $A = \{a_1, a_2, a_3, ...\}$ where each accessibility semantic $a_i = (u, \alpha, \mathbf{l}, \sigma)$ is represented by its semantic type u, semantic confidence α, location vector $\mathbf{l}$, and location accuracy σ. The semantic type u is either: elevator, visually-impaired accessible elevator, visually-impaired accessible signal, curb, staircase, or ramp based on the detector. For the semantic confidence (α), let ℓ be the actual accessibility semantic state and L be its corresponding binary random variable whose states (1/0) denote the semantic being genuine and spurious respectively. Then, α is modeled as the $P(L = 1)$. The accessibility semantic a's 2D location $\mathbf{l} = \begin{bmatrix} x \\ y \end{bmatrix}$ is represented by its latitude (x) and longitude (y) coordinates. The location accuracy σ is an estimate of the location $\mathbf{l}$ accuracy in meters as the radius of a circle centered around the location coordinates.

A detected accessibility semantic (s) from the *Accessibility Semantics Detector Module* will be the input to the *Accessibility Map Construction Module*. The input vector will be $s = (\hat{u}, \hat{\alpha}, \hat{\mathbf{l}}, \hat{\sigma})$ where $\hat{u}$ is the accessibility detector type; i.e. elevator, visually-impaired accessible elevator, visually-impaired accessible signal, curb, staircase, or ramp; $\hat{\alpha}$ is the detector recall value; $\hat{\mathbf{l}}$ is the individual's latitude and longitude location stamp when encountering the semantic; and $\hat{\sigma}$ is the individual's location accuracy estimate.

6.3 Accessibility Map Association

For each detected accessibility semantic s, we assess the likelihood that it was generated by one of the accessibility features in the map A or from a new unobserved one. To estimate the probability of associating s with $a_i \in A$ ($p(s|a_i)$), where $\hat{u} = u$, we consider the sensors' noise and their resultant inherent errors in both the accessibility semantic detection and its location stamp as follows:

$$p(s|a_i) = \eta p(\hat{u}|u) \frac{1}{\sqrt{2\pi}\hat{\sigma}} e^{-\frac{dist(\hat{\mathbf{l}},\mathbf{l})^2}{2\hat{\sigma}^2}} \tag{6.1}$$

Where η is a normalization factor, $p(\hat{u}|u)$ models the uncertainty in the accessibility detection, and the uncertainty in the location is modeled as a Gaussian distribution with $dist(\hat{\mathbf{l}}, \mathbf{l})$ as the geodesic distance between the detected accessibility semantic location ($\hat{\mathbf{l}}$) and map feature a_i location ($\mathbf{l}$). $p(\hat{u}|u)$ is modeled by the map semantic a_i's recall value.

Finally, we model the probability of s originating from an unobserved map feature (a_0) as follows:

$$p(s|a_0) = \eta p_{new} p(\hat{u}|u) \tag{6.2}$$

Where p_{new} is a system parameter for the likelihood of observing a new map feature which is determined empirically. $p(\hat{u}|u)$ is modeled by u's detector recall value. Finally, we associate s with the most probable semantic.

6.4 Accessibility Map Update

Based on the detected accessibility semantic s's association, we update the map accordingly. If s is associated with map feature $a \in A$: We update a using s as follows: First, we update our confidence in the map feature α using Bayes's rule:

$$\alpha = P(L|s_1^n) = \frac{P(s_n|s_1, s_2, ..., s_{n-1}, L)P(L|s_1, s_2, ..., s_{n-1})}{P(s_n|s_1, s_2, ...s_n)} \tag{6.3}$$

Using Markov assumption, we can further simplify it to:

$$\alpha = \frac{P(s_n|L)P(L|s_1, s_2, ..., s_{n-1})}{P(s_n|s_1, s_2, ...s_n)} \tag{6.4}$$

$P(s_n|L)$ is the detector's recall, $P(L|s_1, s_2, ..., s_{n-1})$ is the recursive term and $P(s_n|s_1, s_2, ...s_n)$ is the normalization factor. For the initial probability, we start with a uniform prior. Note that, individuals can suffer from disabilities at different levels. Thus, they can have varying preference on the map accuracy tolerance. The confidence α can help filter erroneous semantics in the map.

Additionally, we update the map feature location $\mathbf{l}$ and its accuracy σ using standard Kalman Filter (KF) formulas [84]. Specifically, we define the KF state as the 2D location $\mathbf{l}$, we assume $\mathbf{l}$ is constant as a map feature location is fixed. Thus, the state prediction ($\mathbf{l}^-$) and its variance (v) are estimated as follows:

$$\mathbf{l}^- = \mathbf{l} \tag{6.5}$$

$$v^- = v \tag{6.6}$$

Then, using s, the KF updates the map feature as follows:

$$k = \frac{v^-}{v^- + \hat{\sigma}^2} \tag{6.7}$$

$$\mathbf{l} = \mathbf{l}^- + K(\hat{\mathbf{l}} - \mathbf{l}^-) \tag{6.8}$$

$$v = (1 - K)v^- \tag{6.9}$$

Where K is the Kalman gain for map-feature a.

On the other hand, if the detected accessibility semantic s is a new map semantic (i.e. not associated with any $a \in A$), we add it to our map ($A = \{a_1, a_2, ..., a_m, a_{m+1}\}$): where a_{m+1} is modeled as follows:

$$u = \hat{u} \tag{6.10}$$

$$\mathbf{l} = \hat{\mathbf{l}} \tag{6.11}$$

$$\alpha = \hat{\alpha} \tag{6.12}$$

$$v = \hat{\sigma}^2 \tag{6.13}$$

That is, we set the new map feature characteristics as the detected accessibility semantic.

6.5 Summary

AccessMap employs a probabilistic framework to construct and update the accessibility map with the crowdsourced data. The framework takes the uncertainty in the detected semantics and the location accuracy into account. This helps improve the map accuracy and speed up the map construction as we show in Chapter 7.

Chapter 7: AccessMap Performance Evaluation

7.1 Overview

We implemented the *AccessMap* system on different Android mobile devices including: LG Nexus 5X, Huawei Mate 9, HTC Nexus One, Samsung Galaxy Nexus, Samsung Galaxy S7, and Samsung Galaxy S Plus. We carried out experiments in the state of Maryland, USA; and the city of Alexandria, Egypt. Thirteen subjects collected a total of 42Km traces with a total of 592 accessibility semantics. Table 7.1 summarizes the accessibility semantics size in our testbeds. The ground-truth for the accessibility semantics were marked manually by the subjects.

Accessibility Semantic Type	Size
Stairs	192
Elevators	78
Signals	72
Curbs	114
Ramps	136
Total	592

Table 7.1: Summary of the test-beds' accessibility semantics characteristics.

For the rest of this chapter, we first show *AccessMap*'s accessibility semantics detectors' performance. Then, we show the effect of the proposed sensor fusion on

the semantics' detection performance and the mobile devices' resources utilization. Thereafter, we show the performance of *AccessMap*'s map construction crowdsourcing algorithm. Finally, we show the system's power consumption.

7.2 Accessibility Detection Performance

We evaluated the accessibility detectors using 10 folds cross validation. Figure 7.1 shows each accessibility semantic detection average precision and recall.

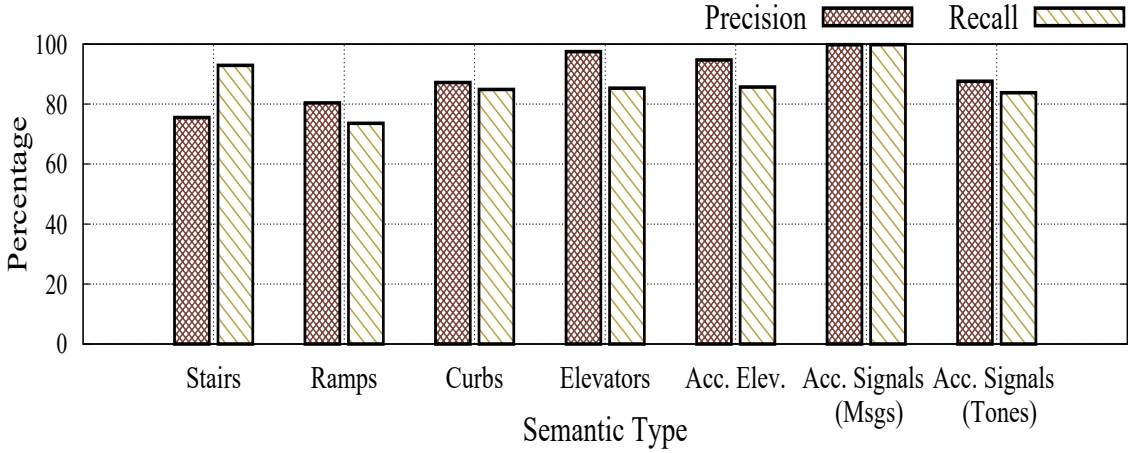
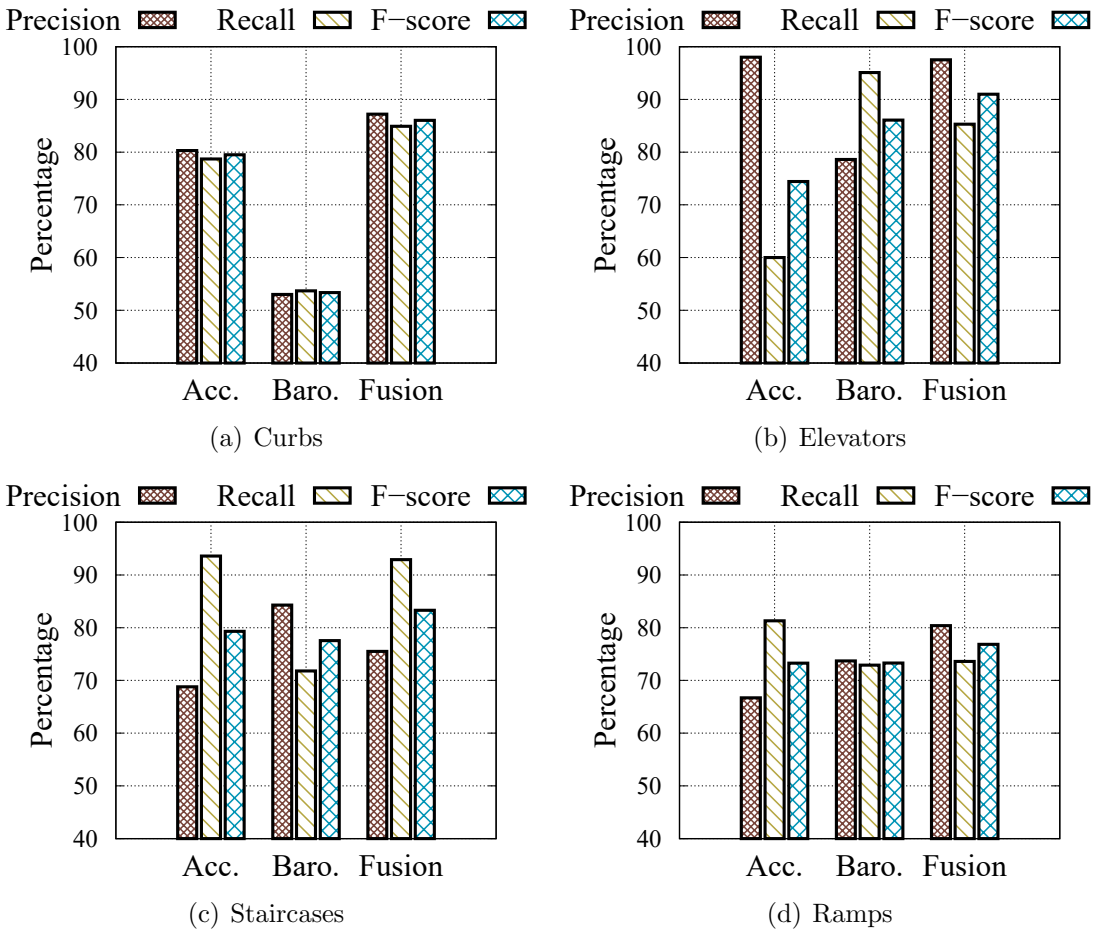


Figure 7.1: The accessibility semantics detection average precision and recall.

Overall, all detectors gave high precision and recall values (typically more than 80%). We note that detecting *ramps* was more difficult than other semantics due to the sensors noise and the fact that ramps are designed to be as shallow as possible making them more similar to walking on a flat surface. On the other hand, both taking a staircase/elevator or stepping up/down from a curb had more robust and stronger signature leading to a higher precision and recall as we see in the figure. Additionally, using the deep neural network for recognizing the audio announcements in elevators and signals had a high precision and recall. Yet, interestingly,

the signal messages were better detected than the elevator announcements due to the clearer/louder signal messages in the street pedestrian signals and the indoor noise affecting the elevators' announcements. Also, we could see that the signal messages were better detected than the signals' tones as tones were more affected by the traffic noise.

7.3 Sensor Fusion and Detection Performance



(a) Curbs

(b) Elevators

(c) Staircases

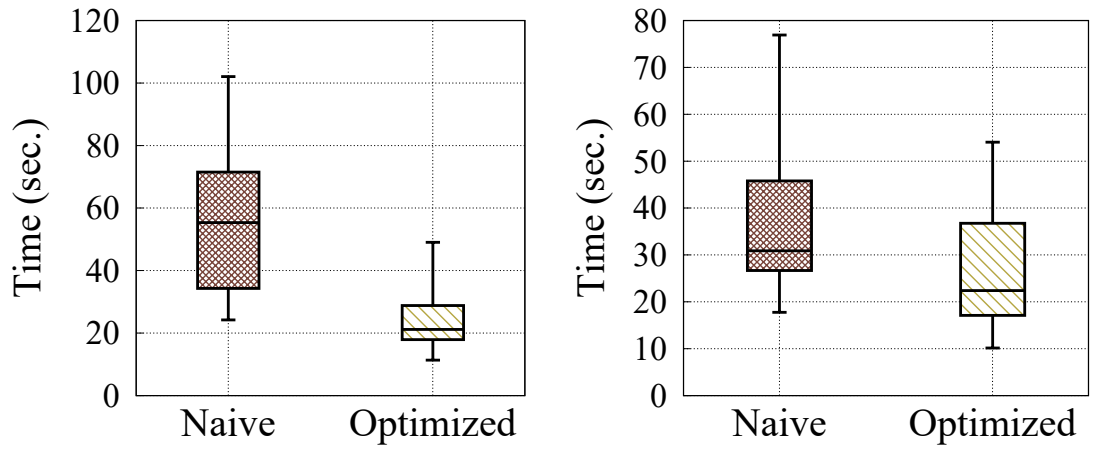
(d) Ramps

Figure 7.2: Effect of fusing the accelerometer and barometer sensors on the accessibility semantics detection performance.

Figure 7.2 compares the accessibility semantics detectors performance (quantified by the precision, recall and F-score measures) to their performance when relying on one of the sensors solely. We evaluated the accessibility detectors performance with the different sensors combinations using 10 folds cross validation. Generally, using the accelerometer sensor only, we can still detect the semantics with reasonable accuracy. This can help *AccessMap* increase its range of supported devices to low-end smartphones that do not typically include the barometer sensor. Yet, the accelerometer and barometer sensors fusion leads to improvements in the overall accessibility detection performance, manifested in a higher F-score for all semantics. For example, while using accelerometer to detect elevators can have high precision because of its distinct pattern (Section 5.2), a fixed segmentation can lead to missing the pattern resulting in a low recall (60%). The proposed dynamic segmentation (through fusing the barometer and accelerometer) helps solving the problem which leads to a significant improvement in the recall by 58.5% and a higher overall F-score by 22.3%. Additionally, as seen in the figure, using the barometer only to detect the elevators, leads to reduction in the precision by 24.1% and 5.9% in the overall F-score relative to the fusion performance.

7.4 Sensor Fusion and Resources Utilization

Figure 7.3 shows the effect of using the accelerometer to optimize the time the microphone is on for pedestrian signals and elevators.



(a) Pedestrian signals

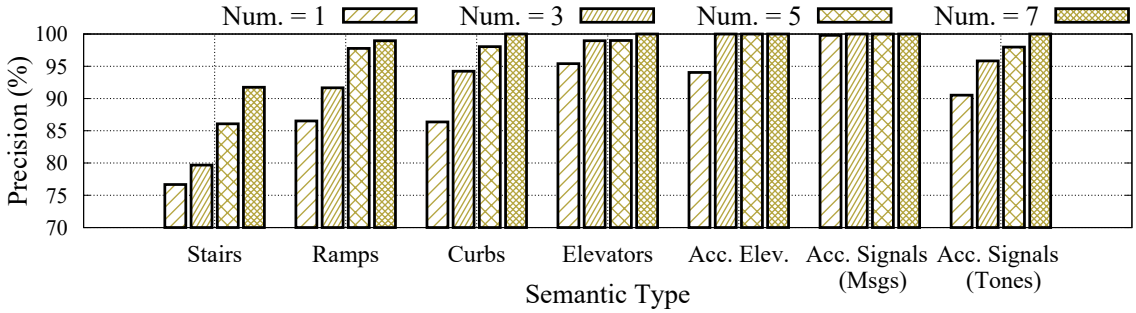
(b) Elevators

Figure 7.3: Effect of using the accelerometer sensor to optimize the time the microphone is turned on for pedestrian signals and elevators.

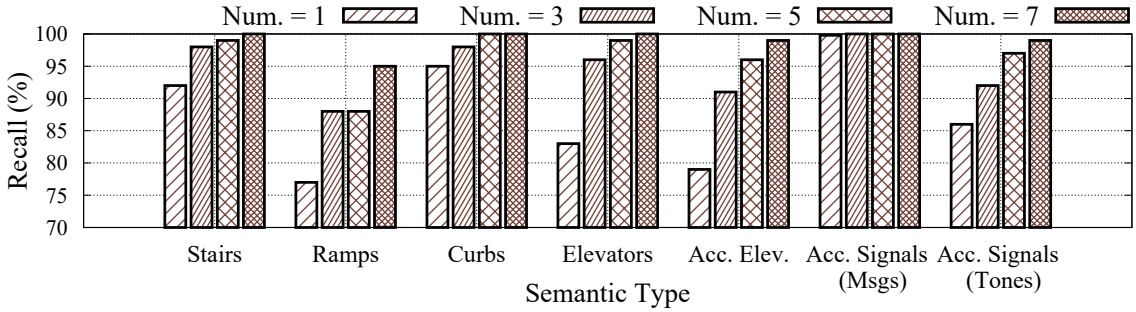
We can see that the our proposed optimization method helped reduce the time the microphone is on by 61.8% for pedestrian signals and 27.4% for elevators. Minimizing the time the microphone is on, reduces *AccessMap*'s privacy sensitivity and reduce the audio data size that needs to be processed. Moreover, the optimization also helps reduce *AccessMap*'s battery consumption when using the microphone as we show in Section 7.7.

7.5 Accessibility Map Semantics Accuracy

As more users come across an accessibility semantic, our map update algorithm incrementally refines the map. Figure 7.4 shows the accessibility map average precision and recall for the different accessibility semantics as we get more encounters data.



(a) Precision



(b) Recall

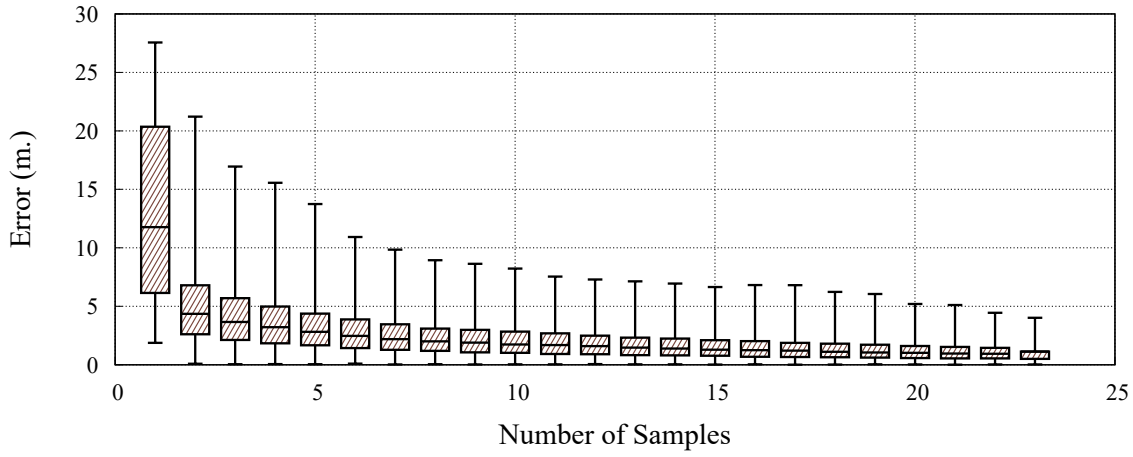
Figure 7.4: Effect of the crowdsourcing on the accessibility map average precision and recall for the different semantics.

We can see that, *AccessMap* can reach more than 95% on average precision and recall for almost all semantics with as few as 5 encounters. This increases to 98.7% average precision and 99% average recall with as few as 7 encounters per space. This rapid incremental refinement is owed to the proposed detectors high accuracy (as shown in Section 7.2) and the crowdsourcing approach.

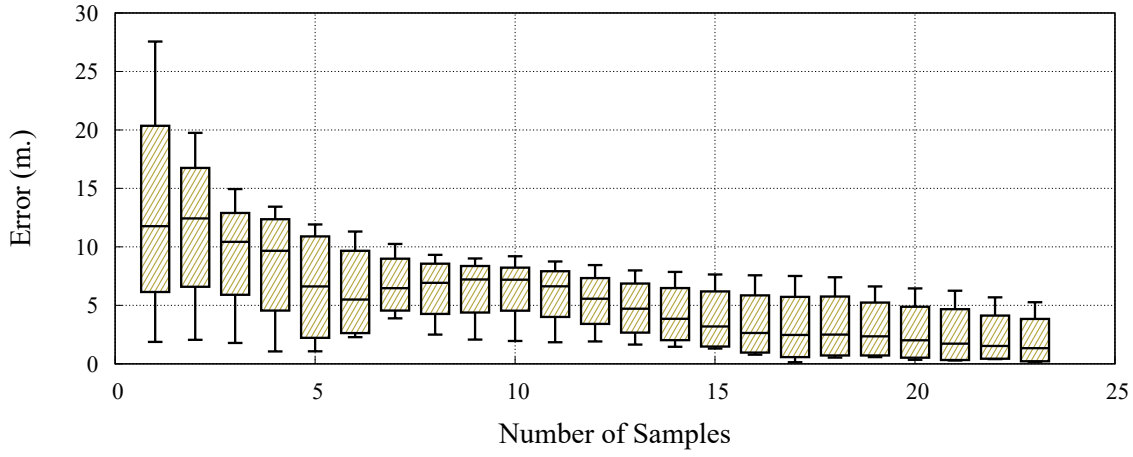
7.6 Accessibility Map Localization Accuracy

AccessMap incrementally refines the accessibility semantics' locations. As more users come across a semantic, its location gets updated using our presented KF-based algorithm. Figure 7.5 shows the accessibility map's semantics locations

accuracy as we get more crowdsourced data, using the KF-based algorithm versus using an average filter .



(a) Kalman Filter



(b) Average Filter

Figure 7.5: Effect of *AccessMap*'s FK-based map update on the map's semantics localization accuracy as compared to using average filter.

We can see that the proposed KF-based map update method helps *AccessMap* reduce the accessibility semantics location error to less than 2m within 10 encounters. Also, as our proposed method takes the uncertainty in the sensor measurements, it improves the map-semantic location estimation by 73.4% over using average filter.

7.7 Power Consumption

To profile *AccessMap*'s power usage, we analyzed its battery consumption based on the sensors used. We ran 10 experiments for each different setting: *AccessMap* (inertial sensors only); *AccessMap* with GPS and WiFi for location; *AccessMap* with GPS and WiFi for location and microphone; *AccessMap* with GPS and WiFi for location and microphone optimized. We did the experiments on LG Nexus 5X and Samsung Galaxy S7 and each experiment was 20 minutes on average. The battery consumption was reported by the Android API. Figure 7.6 shows the experiments results.

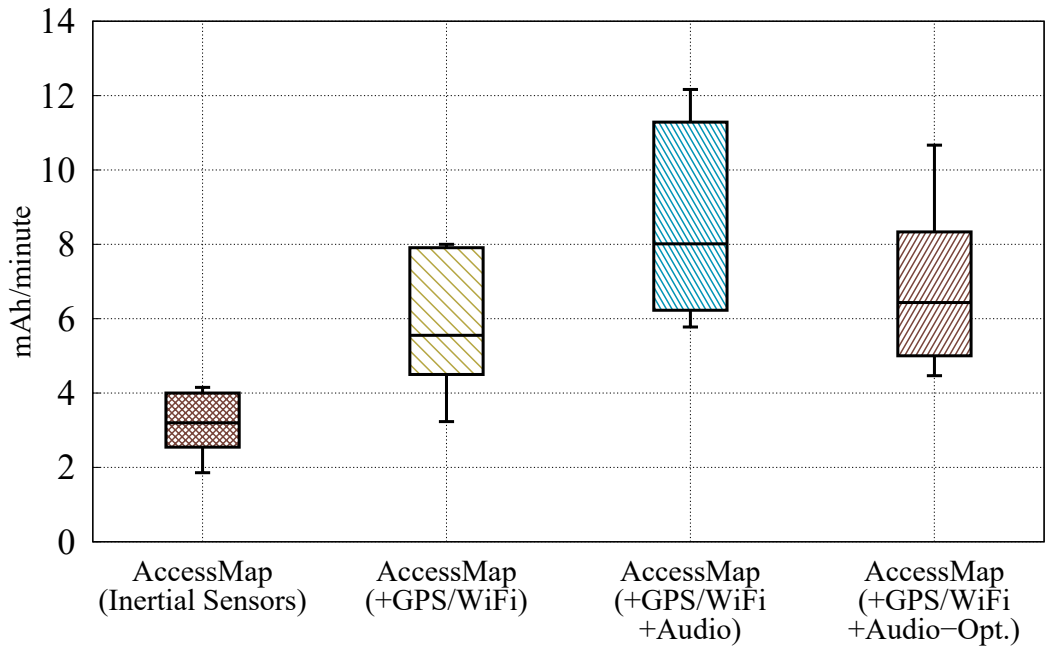


Figure 7.6: *AccessMap* battery consumption based on sensors used.

We can see *AccessMap* inertial-sensor crowdsourcing has a low power consumption (median of 3 mah/minute). Using GPS/WiFi to estimate the location

increases the median power consumption to 5 mah/minute. *AccessMap* low power consumption asserts its viability as a passive crowdsensing system. Additionally, we can see that our optimized microphone usage reduces the battery consumption by 30.7%.

7.8 Summary

In this chapter, we discussed *AccessMap*'s implementation and evaluation. *AccessMap* was implemented on different Android smartphone devices and tested in two different countries. Thirteen volunteer subjects collected a total of 592 accessibility semantics. Our evaluation results showed that the proposed accessibility semantics detectors have a high precision and recall. Additionally, while our proposed sensor-fusion method improves the detectors' performance over using the sensors independently, for low-end phones, which only have the accelerometer sensor, our proposed methods were still able to detect the accessibility semantics with good accuracy. The sensor fusion also helps minimizing the usage of the microphone sensor, leading to better resources utilization. Moreover, *AccessMap*'s crowdsourcing algorithm helps further improve the map thoroughness and the semantics' locations accuracy.

Chapter 8: Conclusion

In this dissertation, we presented the design, implementation and evaluation of the *AccessMap* system. *AccessMap* automatically builds a ubiquitous accessibility map where indoor and outdoor spaces are marked with their accessibility level for the different disability types.

While there has been recent efforts to add accessibility information to digital maps, available systems mainly relied on a manual and/or semi-manual assessment of outdoor spaces where for example, human volunteers take photos of accessibility issues [8]; or Amazon Mechanical Turk crowd-source workers mark and/or validate accessibility issues on Google Street View images [7, 9]. While relying on Google Street Views removes an overhead on volunteers and/or crowdsource-workers to visit the space and to take photos of the space, which leads for speedup in the accessibility map construction. Yet, as our study shows, relying on Google Street View images limit the systems to visual accessibility measures only, limit the generated accessibility maps to outdoor spaces only, and limit the map update rate to the Google Street Views update rate which can vary significantly from an area to another. Finally, Google Street Views gives an image per space which limits the accuracy of the system and leads to relying on manual verification stages to further

improve the accuracy. On the other hand, *AccessMap* proposes a passive crowdsourcing approach to tackle the problem, automatically build the accessibility map and verify its accuracy.

AccessMap collects time and location stamped sensor measurements from the user smartphone devices. Then, via a modular approach, the sensor measurements are processed to detect various accessibility semantics. We define accessibility semantics as either accessibility issues or measures. First, for accessibility issues, those are space features that make it hard for a disabled individual to access that space. For example, a curb in a sidewalk that makes it hard for a wheel-chaired individual to transition from/to the sidewalk. Second, for accessibility measures, those are measures deployed in spaces to assess disabled individuals to use that space independently. For example, installing elevators in foot-bridges to help wheel-chaired individuals use the bridge and cross the road. We presented signal processing and machine-learning based methods to analyze the crowdsourced sensor measurements and detect various accessibility semantics. We have shown that based on the accessibility semantic type, different sensors can capture the user interaction with the semantic using different machine-learning techniques. While we could detect some semantics like staircases with classical supervised machine-learning techniques, detecting audio-messages required more advanced techniques like deep-learning based neural networks. We were able to detect the various semantics with, on average, 89.8% precision and 86.3% recall. To build the accessibility map and assess the space's accessibility state, we present a probabilistic framework that takes into account the uncertainties in the location sensor measurements and the accessibility

semantics detection. We have shown that the crowdsourcing approach increases the average precision and recall to 98.7% and 99% respectively with as few as seven encounters per space. We have also studied the effect of crowdsourcing the different sensors on the user's battery consumption.

Additionally, we have shown how sensor fusion can help reduce privacy- and battery- sensitive sensors usage. For example, we have shown how the accelerometer sensor can be used to control the usage of the microphone sensor for detecting audio-hints to help visually-impaired individuals. Our evaluation shows we could reduce the time the microphone is on by up to 61.8% and reduce the battery consumption by 30.7%. We believe leveraging energy-efficient and less-intrusive sensors, like inertial sensors, to control usage of energy-hungry sensors and/or privacy-sensitive ones, such as cameras and microphones, is a key to their adoption in crowdsourcing systems.

All in all, the *AccessMap* system achieved its main goals of:

- Automatic accessibility map construction: Through the proposed passive crowdsourcing approach, where *AccessMap* collects the spatio-temporal sensor measurements from the user mobile devices. Then, machine learning techniques are employed to identify the space's semantics and rate its accessibility without any human interference.

- Disability coverage scalability: *AccessMap* covered two disability types: mobility disability and vision-impairment. Moreover, with its modular accessibility detection architecture, additional disability types can be supported by just

adding new related semantics and their accessibility assessment rules.

- Spatial coverage scalability: *AccessMap* collects the time- and location- stamped sensor measurements and analyzes them to detect the semantics anywhere indoors and outdoors.

- Accuracy: *AccessMap* has high detection accuracy for the different accessibility semantics. Furthermore, the crowdsourcing approach helps further increase the accuracy as we get more crowdsourced data.

AccessMap's spatial coverage indoors is dependent on the location tag that comes with the sensors. To address this requirement, we presented *Hapi* as a 2.5D WiFi-based indoor localization system. It estimates the user's floor-number and 2D location on that floor which makes it suitable for providing location tags for passive crowdsensing systems. We have presented the design, implementation and evaluation of the *Hapi* system. While there has been lots of research efforts for WiFi-based indoor localization, existing systems either require a tedious and time consuming calibration phase which increases the approaches deployment and maintenance cost, or trades accuracy for ease of deployment.

Hapi balances deployment cost and accuracy. It is built on the observation that a user standing in a floor will observe accesspoints from a number floors above and below her current floor. Yet, one gets to observe more distant accesspoints from closer floors as compared to distant ones. Thus, using accesspoints from other floors can increase the density of accesspoints closer to the user which helps further increase the localization accuracy.

To estimate the user floor, we proposed a novel deep-learning based method. Our technique is independent of the building height; i.e. it can be deployed in any high-rise building. Additionally, we proposed a data augmentation method that helped train the neural network from the WiFi data even without requiring any training data from the deployment building. We believe the data augmentation method can be used to train other RF deep-learning based methods. Our evaluation results show that *Hapi* could achieve 95% exact floor estimation in high-rise buildings.

Taking the floor level estimate into account, *Hapi* estimates the user location PDF over her current floor. To estimate the PDF, we used accesspoints from all floors by extendeding the wall-attenuation factoring model emplyed in RF systems [52] to the Floor-attenuation factoring model. In our evaluation results, we showed how this model helped us improve the 2D location estimation. Additionally, we proposed a novel RSS-Rank probabilistic method. The RSS-Rank method reduces *Hapi*'s sensitivity to heterogeneity. Our evaluation results showed that *Hapi* had a median accuracy of 3.5m with significant improvements over state-of-the-art systems.

Altogether, the *Hapi* system achieved its main goals of:

- Zero calibration deployment overhead: To deploy *Hapi* in a new building, only the building's floorplans and its accesspoints 2.5D locations are required. This improves its scalability in terms of buildings coverage.

- High floor detection accuracy: *Hapi*'s high exact floor detection accuracy im-

proves its usability for location-based services. In particular, for passive crowd-sensing systems such as *AccessMap* where you need to map the accessibility semantic to its floor.

- High 2D localization accuracy: Similarly, *Hapi*'s location estimation accuracy improves its usability and increases its location-based services support range.

8.1 Future Research Directions

Our experience suggests several directions for future work.

8.1.1 Accessible Maps

Realizing digital-maps includes representation, visualization and design. *AccessMap*'s generated accessibility digital maps have to be tailored to two different user types: (1) *direct* users who need to explore the map and (2) *indirect* users who use the accessibility information as a service (e.g. navigation apps). For the first type of users, adding enormous different semantics with various accessibility assessments for different disability types can lead to more cluttered and incomprehensible maps. HCI research studies will need to be conducted to verify suitable accessible visualization methods. Additionally, HCI studies are required to provide the accessibility information in accessible formats based on the disability type for example for visually-impaired individuals.

8.1.2 Disabilities Coverage

AccessMap in its current version supports mobility and vision impairments. However, its modular architecture allows easily supporting other disability types. Further research work is required to build passive-crowdsensing machine-learning algorithms to fuse the sensor measurements and detect other disabilities related accessibility-semantics robustly and accurately.

8.1.3 Sensor-rich Wearable Devices

AccessMap passively crowdosurced spatio-temporal sensor measurements from users smartphone devices. However, other mobile devices like smartwatches and wearable can also be used to detect a wide range of accessibility semantics. Further research work is required to build passive-crowdsourcing machine-learning algorithms using sensor measurements from such wearable devices to detect the accessibility information robustly and accurately.

8.1.4 Temporary Accessibility Issues

AccessMap builds accessibility maps through enriching the digital maps with accessibility related fixed semantics. We identified semantics as either a feature deployed in the space that makes it hard for a disabled person to access that space or a measure deployed in the space to help a disabled one. These semantics are fixed parts of the space. However, temporary issues can occur such as an elevator out of service, or as an event leading to blocking part of a sidewalk, etc. Such temporary

issues affecting a space accessibility was out of the scope of *AccessMap*. Building services to detect and disseminate such information and validate its integrity quickly is a future research direction.

8.1.5 Indoor Localization Deployment

To deploy *Hapi* in a building, the system requires the building's floor plans and its access-points 2.5D location. Relaxing these assumptions is a possible research direction to further reduce the deployment requirements.

Bibliography

[1] Americans with disabilities: 2014. `https://www.census.gov/library/publications/2018/demo/p70-152.html`, Online; accessed May 2018.

[2] Architectural, U.S., and Transportation Barriers. Americans with disabilities act accessibility guidelines for buildings and facilities. 1991.

[3] Heba Aly, Moustafa Youssef, and Ashok Agrawala. Towards ubiquitous accessibility digital maps for smart cities. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2017.

[4] Heba Aly, Anas Basalamah, and Moustafa Youssef. Map++: A crowd-sensing system for automatic map semantics identification. In *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 546–554. IEEE, 2014.

[5] Heba Aly, Anas Basalamah, and Moustafa Youssef. Automatic rich map semantics identification through smartphone-based crowd-sensing. *IEEE Transactions on Mobile Computing*, 16(10):2712–2725, 2016.

[6] Yin Wang, Xuemei Liu, Hong Wei, George Forman, Chao Chen, and Yanmin Zhu. CrowdAtlas: Self-updating maps for cloud and personal use. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 27–40. ACM, 2013.

[7] Kotaro Hara, Vicki Le, and Jon Froehlich. Combining crowdsourcing and google street view to identify street-level accessibility problems. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 631–640. ACM, 2013.

[8] Kelly Shigeno, Sergio Borger, Diego Gallo, Ricardo Herrmann, Mateus Molinaro, Carlos Cardonha, Fernando Koch, and Priscilla Avegliano. Citizen sensing for collaborative construction of accessibility maps. In *Proceedings of the*

10th International Cross-Disciplinary Conference on Web Accessibility, page 24. ACM, 2013.

[9] Kotaro Hara, Jin Sun, Robert Moore, David Jacobs, and Jon Froehlich. Tohme: detecting curb ramps in google street view using crowdsourcing, computer vision, and machine learning. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 189–204. ACM, 2014.

[10] Jin Sun and David W Jacobs. Seeing what is not there: Learning context to determine where objects are missing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5716–5724, 2017.

[11] Dragan Ahmetovic, Roberto Manduchi, James M Coughlan, and Sergio Mascetti. Zebra crossing spotter: Automatic population of spatial databases for increased safety of blind travelers. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, pages 251–258. ACM, 2015.

[12] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google street view: Capturing the world at street level. *Computer*, 43(6):32–38, 2010.

[13] Yin Wang, Hong Wei, and George Forman. Mining large-scale GPS streams for connectivity refinement of road maps. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 438–441. ACM, 2013.

[14] Ye Ding, Jiangchuan Zheng, Haoyu Tan, Wuman Luo, and Lionel M Ni. Inferring road type in crowdsourced map services. In *Database Systems for Advanced Applications*, pages 392–406. Springer, 2014.

[15] Christian Ruhhammer, Michael Baumann, Valentin Protschky, Horst Kloeden, Felix Klanner, and Christoph Stiller. Automated intersection mapping from crowd trajectory data. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):666–677, 2017.

[16] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The Pothole Patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 29–39. ACM, 2008.

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

[18] Qianru Wang, Bin Guo, Leye Wang, Tong Xin, He Du, Huihui Chen, and Zhiwen Yu. Crowdwatch: dynamic sidewalk obstacle detection using mobile crowd sensing. *IEEE Internet of Things Journal*, 4(6):2159–2171, 2017.

[19] Heba Aly, Anas Basalamah, and Moustafa Youssef. Automatic rich map semantics identification through smartphone-based crowd-sensing. *IEEE Transactions on Mobile Computing*, 16(10):2712–2725, 2017.

[20] Moustafa Alzantot and Moustafa Youssef. Crowdinside: automatic construction of indoor floorplans. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 99–108. ACM, 2012.

[21] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and James Collins. Global positioning system. theory and practice. *Global Positioning System. Theory and practice., by Hofmann-Wellenhof, B.; Lichtenegger, H.; Collins, J.. Springer, Wien (Austria), 1993, 347 p., ISBN 3-211-82477-4, Price DM 79.00. ISBN 0-387-82477-4 (USA).*, 1, 1993.

[22] Youjing Cui and Shuzhi Sam Ge. Autonomous vehicle positioning with GPS in urban canyon environments. *IEEE Transactions on Robotics and Automation*, 19(1):15–25, 2003.

[23] C O'Driscoll, G Lachapelle, and M E Tamazin. Investigation of the benefits of combined GPS/GLONASS receivers in urban environments. In *Proc. on RIN NAV10 Conf. on Position, Location, Timing: Everyone, Everything, Everywhere*, 2010.

[24] Google's MyLocation. `http://www.google.com/mobile/maps/`, Online; accessed January 2015.

[25] Mohamed Ibrahim and Moustafa Youssef. CellSense: A probabilistic RSSI-based GSM positioning system. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE, 2010.

[26] Mohamed Ibrahim and Moustafa Youssef. Cellsense: An accurate energy-efficient GSM positioning system. *Vehicular Technology, IEEE Transactions on*, 61(1):286–296, 2012.

[27] Mohamed Ibrahim and Moustafa Youssef. A hidden markov model for localization using low-end GSM cell phones. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.

[28] Yu-Chung Cheng, Yatin Chawathe, Anthony LaMarca, and John Krumm. Accuracy characterization for metropolitan-scale Wi-Fi localization. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 233–245. ACM, 2005.

[29] Skyhook wireless. http://www.skyhookwireless.com/, Online; accessed January 2015.

[30] Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter,

et al. Place lab: Device positioning using radio beacons in the wild. In *Pervasive computing*, pages 116–133. Springer, 2005.

[31] Ionut Constandache, Shravan Gaonkar, Matt Sayler, Romit Roy Choudhury, and Landon Cox. EnLoc: Energy-efficient localization for mobile phones. In *INFOCOM 2009*, pages 2716–2720. IEEE, 2009.

[32] Moustafa Youssef and Ashok Agrawala. Location-clustering techniques for WLAN location determination systems. *International Journal of Computers and Applications*, 28(3):278–284, 2006.

[33] Reham Mohamed, Heba Aly, and Moustafa Youssef. Accurate and efficient map matching for challenging environments. In *Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 401–404. ACM, 2014.

[34] Reham Mohamed, Heba Aly, and Moustafa Youssef. Accurate real-time map matching for challenging environments. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):847–857, 2016.

[35] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. No need to war-drive: Unsupervised indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 197–210. ACM, 2012.

[36] Moustafa Alzantot and Moustafa Youssef. UPTIME: Ubiquitous pedestrian tracking using mobile phones. In *Wireless Communications and Networking Conference (WCNC)*, pages 3204–3209. IEEE, 2012.

[37] Ionut Constandache, R Roy Choudhury, and Injong Rhee. CompAcc: Using mobile phone compasses and accelerometers for localization. In *IEEE INFOCOM*, 2010.

[38] Moustafa Youssef, M Amir Yosef, and Mohamed El-Derini. GAC: Energy-efficient hybrid GPS-accelerometer-compass GSM localization. In *Global Telecommunications Conference (GLOBECOM 2010)*, pages 1–5. IEEE, 2010.

[39] Heba Aly and Moustafa Youssef. Dejavu: an accurate energy-efficient outdoor localization system. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 154–163. ACM, 2013.

[40] Paramvir Bahl, Venkata N Padmanabhan, Victor Bahl, and Venkat Padmanabhan. RADAR: An in-building rf-based user location and tracking system. In *IEEE INFOCOM*. IEEE, 2000.

[41] AKM Mahtab Hossain and Wee-Seng Soh. A survey of calibration-free indoor positioning systems. *Computer Communications*, 66:1–13, 2015.

[42] Zheng Yang, Chenshu Wu, Zimu Zhou, Xinglin Zhang, Xu Wang, and Yunhao Liu. Mobility increases localizability: A survey on wireless indoor localization using inertial sensors. *ACM Computing Surveys (Csur)*, 47(3):54, 2015.

[43] Moustafa Youssef and Ashok Agrawala. The Horus WLAN location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 205–218. ACM, 2005.

[44] Ville Honkavirta, Tommi Perala, Simo Ali-Loytty, and Robert Piché. A comparative survey of WLAN location fingerprinting methods. In *2009 6th workshop on positioning, navigation and communication*, pages 243–251. IEEE, 2009.

[45] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. No need to war-drive: Unsupervised indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 197–210. ACM, 2012.

[46] 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 Transactions on Mobile Computing*, 15(7):1770–1782, 2016.

[47] Jiang Dong, Yu Xiao, Marius Noreikis, Zhonghong Ou, and Antti Ylä-Jääski. iMoon: Using smartphones for image-based indoor navigation. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 85–97. ACM, 2015.

[48] Ahmed Eleryan, Mohamed Elsabagh, and Moustafa Youssef. AROMA: Automatic generation of radio maps for localization systems. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 372–373. Springer, 2010.

[49] Heba Aly and Moustafa Youssef. An analysis of device-free and device-based WiFi-localization systems. *International Journal of Ambient Computing and Intelligence (IJACI)*, 6(1):1–19, 2014.

[50] Juraj Machaj, Peter Brida, and Robert Piché. Rank based fingerprinting algorithm for indoor positioning. In *2011 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–6. IEEE, 2011.

[51] Ran Liu, Chau Yuen, Jun Zhao, Jindong Guo, Ronghong Mo, Vishesh N Pamadi, and Xiang Liu. Selective AP-sequence based indoor localization without site survey. In *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2016.

[52] Rizanne Elbakly and Moustafa Youssef. A robust zero-calibration RF-based localization system for realistic environments. In *13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2016*, pages 1–9. IEEE, 2016.

[53] Kyeong Soo Kim, Sanghyuk Lee, and Kaizhu Huang. A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on Wi-Fi fingerprinting. *Big Data Analytics*, 3(1):4, 2018.

[54] Giuseppe Caso, Luca De Nardis, Filip Lemic, Vlado Handziski, Adam Wolisz, and Maria-Gabriella Di Benedetto. ViFi: Virtual fingerprinting WiFi-based indoor positioning via multi-wall multi-floor propagation model. *IEEE Transactions on Mobile Computing*, 2019.

[55] COST Action. 231,"digital mobile radio towards future generation systems. Technical report, final report," technical report, European Communities, EUR 18957, 1999.

[56] Preeti Bhargava, Shivsubramani Krishnamoorthy, Aditya Karkada Nakshathri, Matthew Mah, and Ashok Agrawala. Locus: An indoor localization, tracking and navigation system for multi-story buildings using heuristics derived from Wi-Fi signal strength. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 212–223. Springer, 2012.

[57] Preeti Bhargava, Shivsubramani Krishnamoorthy, Anilesh Shrivastava, Aditya Karkada Nakshathri, Matthew Mah, and Ashok Agrawala. Locus: robust and calibration-free indoor localization, tracking and navigation for multi-story buildings. *Journal of location Based services*, 9(3):187–208, 2015.

[58] Rizanne Elbakly, Heba Aly, and Moustafa Youssef. TrueStory: Accurate and robust rf-based floor estimation for challenging indoor environments. *IEEE Sensors Journal*, 18(24):10115–10124, 2018.

[59] Zheng Yang, Zimu Zhou, and Yunhao Liu. From RSSI to CSI: Indoor localization via channel response. *ACM Computing Surveys (CSUR)*, 46(2):25, 2013.

[60] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. SpotFi: Decimeter level localization using WiFi. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 269–282. ACM, 2015.

[61] Xuyu Wang, Lingjun Gao, and Shiwen Mao. CSI phase fingerprinting for indoor localization with a deep learning approach. *IEEE Internet of Things Journal*, 3(6):1113–1123, 2016.

[62] Masayuki Murata, Dragan Ahmetovic, Daisuke Sato, Hironobu Takagi, Kris M Kitani, and Chieko Asakawa. Smartphone-based indoor localization for blind navigation across building complexes. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2018.

[63] Heba Aly, Anas Basalamah, and Moustafa Youssef. Robust and ubiquitous smartphone-based lane detection. *Pervasive and Mobile Computing*, 26:35–56, 2016.

[64] Heba Aly, Anas Basalamah, and Moustafa Youssef. Lanequest: An accurate and energy-efficient lane detection system. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 163–171. IEEE, 2015.

[65] Moustafa Youssef. Towards truly ubiquitous indoor localization on a worldwide scale. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 12. ACM, 2015.

[66] Suining He, S-H Gary Chan, Lei Yu, and Ning Liu. SLAC: Calibration-free pedometer-fingerprint fusion for indoor localization. *IEEE Transactions on Mobile Computing*, 17(5):1176–1189, 2017.

[67] Heba Aly and Ashok Agrawala. Hapi: A robust pseudo-3D calibration-free WiFi-based indoor localization system. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 166–175. ACM, 2018.

[68] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[69] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[70] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[71] Scott Y Seidel and Theodore S Rappaport. 914 MHz path loss prediction models for indoor wireless communications in multifloored buildings. *IEEE transactions on Antennas and Propagation*, 40(2):207–217, 1992.

[72] Heba Aly, Anas Basalamah, and Moustafa Youssef. Accurate and energy-efficient GPS-less outdoor localization. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 3(2):4, 2017.

[73] Mohinder S Grewal. Kalman filtering. In *International Encyclopedia of Statistical Science*. Springer, 2011.

[74] Indoor Maps - Google Maps. `https://www.google.com/maps/about/partners/indoormaps/`, 2018. Online; accessed May 2018.

[75] A.V. Williams Building Maps. `https://clarknet.eng.umd.edu/maps`, 2018. Online; accessed May 2018.

[76] Cambridge Floor Plans. `http://reslife.umd.edu/halls/cambridge/cambridge/floor_plans/floorplan.html`, 2018. Online; accessed May 2018.

[77] Bel Air Floor Plans. http://reslife.umd.edu/halls/cambridge/bel_air/floor_plans/floorplan.html, 2018. Online; accessed May 2018.

[78] Cumberland Hall Floor Plans. http://reslife.umd.edu/halls/cambridge/cumberland/floor_plans/floorplan.html, 2018. Online; accessed May 2018.

[79] Centreville Hall Floor Plans. http://reslife.umd.edu/halls/cambridge/centreville/floor_plans/floorplan.html, 2018. Online; accessed May 2018.

[80] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE, 2018.

[81] Google cloud speech-to-text. https://cloud.google.com/speech-to-text/, Online; accessed March 2019.

[82] Accessible pedestrain signals — a guide to best practices. http://www.apsguide.org/appendix_a_prowag.cfm, Online; accessed April 2019.

[83] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[84] Yaakov Bar-Shalom, Peter K Willett, and Xin Tian. *Tracking and data fusion*. YBS publishing Storrs, CT, USA:, 2011.