Smartphone Localization for Indoor Pedestrian Navigation

Vivek Roy

CMU-RI-TR-22-23

April 2022

The Robotics Institute

Carnegie Mellon University

Pittsburgh, PA

Thesis Committee:

Prof. Kris Kitani, *chair*

Prof. Michael Kaess

Akash Sharma

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Global positioning system (GPS) interfacing with applications such as Google Maps has proven very useful for navigation in outdoor open settings. However in crowded metropolitan environments with high rise buildings or in indoor settings, GPS quickly becomes unreliable. Using sensors found on commodity smartphones to perform accurate pedestrian localization in complex indoor settings remains a challenging problem. RSSI based methods provide absolute positioning but provide sparse updates and require a dense network of beacons. Inertial Measurement Unit (IMU) based methods provide high rate, smooth, relative positioning but are very sensitive to noise and drift with time. In this thesis we propose a smartphone based indoor localization system which combines the absolute positioning from Bluetooth RSSI based localization with the smooth relative positioning from IMU based inertial odometry and map information to fix drift. This system reduces the number of beacons required for accurate localization by a factor of 5 compared to state-of-the-art RSSI only methods. Compared to state-of-the-art IMU only methods, our method is 2.4x more accurate and compared to state-of-the-art inertial map-prior networks it is 19% more accurate. We deployed the deep models to mobile phones in the form of iOS and Android apps to run real-time localization thereby facilitating path planning and turn-by-turn navigation.

Acknowledgments

I would like to start by thanking my advisor, Prof. Kris Kitani, for giving me an opportunity to work on this project. His guidance and unwavering support has helped me a lot to grow as a researcher. This project would not be possible without the weekly one-on-one meetings and all the insightful advice he provided.

Furthermore, I am also extremely grateful to all the members of KLab, both past and present. The research environment they provide along with a sounding board for ideas was indispensable. I would like to particularly thank Karnik Ram, Sean Crane, Harsh Agarwal, Scott Sun and Dennis Melamed for all their work and support in getting us to the point where this project was possible. I would also like to thank my thesis committee: Prof. Kris Kitani, Prof. Michael Kaess, and Akash Sharma for generously offering their time and feedback on this work.

Lastly, I would like to thank my family and friends. Their support, both academic and emotional, through this challenging period has kept me motivated and complete this work in this short amount of time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Let us start by defining some terminology that will be used extensively throughout this document. *Localization* is the process of determining where a mobile robot is located with respect to its environment. The map of the environment is known or provided. *Mapping* on the other hand is the process of generating the map of the environment given that the location of the robot is known. *Simultaneous Localization and Mapping (SLAM)* is the combination of the two – localization without having a map of the environment. *Navigation*[1] is the combination of localization with path planning, i.e. given a starting point (from localization) and one or more destinations find the "best" path defined by some evaluation metric to compare goodness of two trajectories.

Smartphone based localization has proven to be highly effective and useful in outdoor settings in the past decade. Global Positioning System (GPS) started by the US Department of Defence in the 1970s found widespread use in navigation systems and became commonplace with the advent of the smartphone. Map services such as Google maps and Apple maps running on everyday smartphones use GPS modules built in practically all modern smartphones to provide localization, planning and turn-by-turn navigation. Such services have proven the usefulness of localization

[1]Navigation is a poorly defined term and different people mean different things when they say navigation. This definition of navigation is what we mean throughout this text.

technology to everyone. Unfortunately, GPS requires an unobstructed line-of-sight to four or more GPS satellites. Due to this nature, its use is limited to outdoor settings with little to no obstruction to the GPS satellites. In large metropolitan cities which have high rise buildings it fails to localize accurately even outdoors as the high rises reduce the accurate use of the signal either in the form of not receiving the signal, or, signal being observed indirectly after bouncing off a tall building resulting in "multi-path" measurements. In indoor environments a similar challenge is experience. Researchers have been looking at multiple different sensors to fill this void as localization is an important basic requirement for a lot of robotics and automation tasks.

1.1 Sensors

1.1.1 Camera

Cameras are probably the most common as well as intuitive sensors to use for such a task as eyes are the primary means of localization and mapping used by humans and most of the animal kingdom. To that extent, cameras have been used extensively in SLAM applications be it in unmanned ground vehicles (UGV) or unmanned aerial vehicles (UAV) commonly known variations of which are self driving cars and autonomous drones. With the rapid growth of the field of computer vision and image processing, camera based localization has also seen significant growth both in terms of accuracy as well as efficiency resulting in fast, accurate, real-time systems. In combination with the fact that different kinds of cameras have abilities to see in the dark or combination of multiple cameras can be used to estimate depth, structure, optical flow etc. cameras are used extensively in a wide range of devices and robotics is no exception. All modern day smartphones have very high quality cameras that can and have been leveraged by a lot of smartphone based applications to provide localization and navigation in places where GPS stop working. Inspite of their great success in robotics applications, in the context of smartphone based localization and navigation, camera based approaches have not gained significance due to their

high battery drain and privacy concerns which are only becoming more important to everyday consumers. While recording in outdoor public environments might be acceptable, user studies have shown that indoor private environments are far more concerned about the privacy and security of their environment. Making a lower power consumption and privacy preserving localization system using cameras that are recording their surroundings is a challenging problem to solve that we leave for a different study.

1.1.2 GPS

Global Positioning System (GPS) works by calculating the distance of the receiver (smartphone or robot) from multiple GPS satellites. Each satellite carries an accurate record of its position and time, and transmits that data to the receiver. Since the speed of radio waves is constant and independent of the satellite speed, the time delay between when the satellite transmits a signal and the receiver receives it is proportional to the distance from the satellite to the receiver. At a minimum, four satellites must be in view of the receiver for it to compute four unknown quantities – three position coordinates and the deviation of the receiver clock from satellite time. GPS modules in smartphones are capable of achieving upto 5 meters of accuracy. However this requires unobstructed line of sight to the satellites without which the accuracy drop drastically very quickly. Thus, being surround with tall buildings or being in an indoor environment with the walls obstructing a line of sight to the satellites, localization using GPS is not very useful.

1.1.3 Radar

Radar is a detection system that uses high frequency electromagnetic waves to determine the distance, angle and radial velocity of objects. The system consists of a transmitter and a receiver. Radio waves from the transmitter reflect off the object and return to the receiver, giving information about the object's location and speed. It is widely used in the aviation industry and military equipment for air and terrestrial traffic and flight control systems, air-defense systems, antimissile systems,

aircraft anti-collision systems, guided missile target locating systems. Marine radars are also used in ocean surveillance systems to locate landmarks and other ships. Some self-driving cars are also using radar combined with other sensors for long range visibility. While radar sensors are useful in long range applications and might be helpful to add them to robotics applications with an appropriate power level to achieve a desired level of range, they are not commonly found on smartphones today.

1.1.4 Sonar

Similar to Radar, sound navigation and ranging or Sonar uses sound waves for ranging and navigation. It is mostly used in under water equipment for navigation, measuring distances (ranging), or even communicating with or detecting objects on or under the surface of the water, such as other vessels. Active sonar, like Radar, has a transmitter and receiver, which emits a sound and listens for echos thereby measuring distance using time-of-flight. Passive sonars only has a receiver and is listening for sound made by other objects thereby detecting presence. Active and passive sonars have many military, civilian and scientific applications. In naval warfare, passive sonars are used because of their stealth characteristics to detect presence of other vessels. They are widely used in both surface ships and submarines. Target motion analysis (TMA) can be done to determine trajectory. Torpedoes sometimes have sonar built into them to guide them more accurately to their target. Sonars are used for a lot of scientific applications for biomass detection, wave and water velocity measurements and under water gas leak detection. Under water robotics extensively use sonar for ranging and navigation. In self-driving cars sonar is used for obstacle detection and collision avoidance systems. However, similar to radar, sonar has not been commonly found in modern day smartphones. Some work has been done to use the speakers and microphones in smartphones for doing sound based ranging but many more studies need to be done on this field which we leave for a different time.

1.1.5 LiDAR

Similar to radar and active sonar, LiDAR or light detection and ranging, uses electromagnetic waves that reflect off of objects to calculate time of flight and thus estimate the distance and velocity of objects. LiDAR uses lasers to achieve this, thought the wavelength of the light used depends on the environment – earth's surface, terrestrial or the ocean bottom. LiDAR is commonly used to make high-resolution maps, with applications in surveying, archaeology, geography and every other field that has some use of 3D maps and models. LiDAR is the most common sensor after cameras found in a wide range of robotics applications including but not limited to self-driving cars. A variant of LiDAR – flash LiDAR – are event found on modern smartphones to help in augmented reality (AR) algorithms and applications. LiDAR sensors are a key component in many SLAM applications to generate an accurate map of the environment. However, these kind of sensors are very expensive, consume a lot of battery power and require very high computational capabilities to make the best use of the data. While flash LiDARs might be commonplace in smartphones in a few years, as of this writing they are only found in the most high-end smartphones to serve a very specific need for AR applications.

1.1.6 IMU

Inertial Measurement Units (IMU) are a combination of accelerometer, gyroscope and magnetometer. IMU is used in approaches like Pedestrian Dead Reckoning (PDR) or closed source estimators like the iOS CoreMotion API. IMUs are used for step detection to detect when the user has taken a step. Making an assumptions about the size of the user's stride, when a step is detected, add this stride length in the direction of the estimated heading to the last estimated position to update the estimate. Such approaches experience significant drift, and the stride length assumption in particular is easily broken. More accurate IMU sensors have been developed that do not suffer the drifting and noise problems that are commonplace in IMU sensors found in smartphones (MEMS IMUs). The US Dept. of Defense in particular have developed

highly accurate IMU units which can directly be integrated to get very accurate position estimates in GPS deprived or jammed regions. Such IMUs can cost thousands of dollars and we do not expect such IMUs to become commonplace anytime soon. IMUs are commonly used along with cameras in a Visual Inertial Odometry (VIO) system to help algorithms differentiate camera motion from scene object motion which is a difficult problem in visual only odometry or tracking tasks. IMU sensors being cheap, low power, privacy protecting and found on practically all smartphones can be very useful in making a smartphone based localization system.

1.1.7 RSSI

Received signal strength indicator (RSSI) is a measurement of the power present in a received radio signal. RSSI based techniques using radio signals of different frequencies have been developed. Bluetooth Low Energy (BLE), WiFi, Radio-frequency identification (RFID), Near Field Communication (NFC) and Ultra-wide band (UWB) radios are all common radio signals that are used in RSSI based approaches. RSSI-based localization can be categorized into two categories – fingerprinting and trilateration. Fingerprinting uses a database of RSSI strengths at various known reference points captured as a data collection step. At inference time, this database is cross-referenced to estimate the location. Trilateration uses models of RSSI signal attenuation to form geometric constraints on the receiver with respect to a set of transmitters (or vice versa). Closed source APIs such as Google Fused Location Provider (FLP) uses WiFi RSSI in combination with traditional GPS to provide more accurate and battery efficient geolocation services. They create a map of WiFi access points accross the world and then use fingerprinting approaches to estimate location. The high accuracy of RSSI based localization techniques given a dense enough network of sensors are the primary reason that they have been so commonly used for indoor applications. WiFi and BLE hardware being commonplace in almost all portable electronic devices also support the use of such sensors for a smartphone based localization system.

Chapter 2

Related Work

RSSI only, odometry only, combining RSSI and odometry, combining odometry with maps, combining RSSI and odometry with maps – all possible groupings have been studied in some form for indoor agent localization including the different kinds of aforementioned sensors which are not found in commodity smartphone hardware. In this thesis, we will limit ourselves to the methods that can, potentially, run on a smartphone.

2.1 RSSI only methods

Various RSSI based localization techniques using Radio-frequency identification (RFID) [1] and Ultra-wide band (UWB) radios [2] have been developed as RSSI information is available ubiquitously and is easy to retrieve using smartphone apps. RSSI from Bluetooth Low Energy (BLE) or WiFi remain a popular choice [3, 4, 5, 6] as Bluetooth and WiFi chips are installed on across the board on smartphones unlike NFC or UWB chips which are often omitted on budget offerings for cost cutting.

Deciding between RSSI from BLE or WiFi is a far more difficult one to make. On one hand WiFi is already installed and actively maintained in most buildings for the connectivity requirements. The biggest strength is also its biggest weakness. WiFi scanning can be easily exploited to locate the user without asking for location

permission and due to privacy concerns on iOS WiFi scanning and fingerprinting is not available to third-party apps. Thus, if iOS is an important target platform, WiFi scanning is out of question. On the other hand BLE beacons are low power, usually battery operated devices that require very little configuration and maintenance to be useful as a proximity detection or localization technique. Multiple simple communication protocols have been developed for these dumb devices – iBeacon, Eddystone and Altbeacon. For these reasons and since iOS is by-far the most used platform in the United States, we went with BLE RSSI. However, the low power of BLE beacons limit the range of such beacons and a dense network of these beacons are required for a reasonable level of accuracy.

RSSI-based localization can be categorized into two categories – fingerprinting and trilateration. Fingerprinting uses a database of RSSI strengths at various known reference points captured as a data collection step. At inference time, this database is cross-referenced to estimate the location. Non-parametric methods such as nearest neighbors, as well as parametric approaches such as neural networks have been used to achieve this. Trilateration uses parametric models of RSSI signal attenuation to form geometric constraints on the receiver with respect to a set of transmitters (or vice versa). Trilateration methods [7, 8] are able to estimate position within 2 meters accuracy. Models such as Gaussian models [3], Monte Carlo [9], Bayesian [10], Hidden Markov Models [11], and radio propagation models [12] have all been proposed to achieve this. Ring overlapping approaches have also been proposed [13, 14], however, these methods tend to require additional calibration, both at the antenna and the phone level. This calibration process significantly inhibits their usage and generalization ability across different phone models. Modelling signal propagation methods for complex indoor environments is extremely difficult [15] and lead to wrong estimates.

With the advent of machine learning, methods have tried to leverage these parametric approximators instead. [16, 17, 18, 19] combine use of robust feature estimates from deep networks with classical machine learning techniques using algorithms like probabilistic KNNs and HMMs. [20] proposed a deep network that is trained end-to-end with WiFi RSSI. [21] tried solving the inverse problem of localizing beacon

scanners, using RSSI strengths received.

All these methods require a very dense network of these low power beacons and do not generalize well to a wide range of hardware on the receiving end. Using multiple experiments we show that different phones have very different signal characteristics of the received RSSI and simple signal processing have to be specifically tuned on a per-device basis to make them generalize across a range of devices.

2.2 Odometry only methods

Traditional, non-learning based inertial odometry approaches like Pedestrian Dead Reckoning (PDR) generally begin by estimating device heading via filters like Magwick's [22] or closed source estimators like the iOS CoreMotion API which utilizes gyroscope, accelerometer, and magnetometer signals. It is usually followed by a step detection using the IMU signals to detect when the user has taken a step. PDR methods make assumptions about the size of the user's stride and, when a step is detected, add this stride length in the direction of the estimated heading to the last estimated position to update the estimate [23]. Such approaches experience significant drift, and the stride length assumption in particular is easily broken.

Many recent inertial odometry only methods have been using learning based approaches. [24] uses a deep learning based approach to map IMU measurements to user polar displacements like PDR but without making any assumptions about the user stride thereby getting significant improvements in accuracy. [25] explored residual networks with temporal CNNs and RNNs to find the best architecture for the task of deep inertial localization. [26] added a Kalman filter to this architecture to further improve accuracy. This approach improves heading estimation and achieves even more accurate localization. [27] uses a data driven approach to estimating accurate orientations thereby further reducing inertial heading errors. [28] combine transformer-based neural networks with [25] and a large train dataset to impose stronger constraints on possible walk-able areas in a given environment.

2.3 Odometry + Map information

Odometry only localization methods have gained significant improvements in accuracy by combining them with map information as it helps detect and fix any drift in the estimation by detecting infeasible trajectories against the given map information. Graph based systems build a map of nodes and edges with nodes in all walk-able areas on the map and connecting every pair of nodes using an edge if that edge is a feasible transition, i.e. there are no obstacles on that transition. The user's trajectory is then constructed as a sequence of such physically feasible transitions (edges) in the graph. [29] implements a hidden Markov model (HMM) to localize agents with unstable GPS signals on a graph of outdoor road segments. GPS signals are matched to road segments, and based on known factors about the maximum travel speed of the user and other constraints, a series of maximally likely transitions are estimated via the Viterbi algorithm. [30] uses deep attention networks to outperform HMMs in some scenarios for matching user travel to road segments.

In indoor environments, the graph typically has much less constraints compared to the road network. Fewer constraints combined with higher accuracy requirement leads to a dense graph. Conditional random fields (CRFs) have been traditionally commonly used with such graph based approaches. [31] uses human step length as the space between two nodes when setting up the graph and fully covering the physically accessible space. Inertial odometry is used to transition between the nodes of the graph. Graph based approaches can never localize more precisely than the node spacing parameter and to that extent a pedestrian with slower than average walking speed causes large drifts in estimates. Such approaches are also usually very computationally heavy as they have to have a large densely connected graph in memory and then traverse it.

Bayesian filtering approaches improve localization accuracy by combining information sources like odometry and map information directly without an underlying set of possible states to transition between. Extended Kalman filtering (EKF) is an efficient approach to estimating user state from both odometry and external information. However, it suffers from only maintaining one estimate of state and assumes Gaussian

uncertainty. Maintaining multiple estimates of state until one is proven most likely can be very useful, especially in indoor navigation, such as when it is uncertain which side of a narrow obstacle the user has traveled on.

[32] uses particle filters (PF) as the state distributions are unknown, change over time, and could be multi-modal. They maintain an estimated state distribution by storing a set of particles which discretely represent the distribution. Odometry is used to propagate the estimates (particles). A small amount of noise is artificially added to the odometry to simulate the uncertainty in measurement. Particles are then weighted based on external information such as how well their state matches the output from a LiDAR scan. A resampling process keeps the particles with probability proportional to their weight, maintaining a discrete distribution which best combines both odometry and external information. Particle filters are simple and adaptable, lending themselves well to localization on low compute devices.

Map constraints with particle filters are usually incorporated using heuristics which heavily downweight the probability of particles with infeasible odometry propagation according to the map. [23][33][34] have shown the practical effectiveness of particle filter based approaches, but down-weighting is very sensitive to odometry noise and can lead to particle weight degeneracy. [35] uses particle filters to calculate overlap between map obstacles and a computed trajectory found by integrating a few previous odometry measurements backwards from a particle location. Particles are given more weight if there are fewer overlaps. This type of multi-step heuristic is less sensitive to odometry noise than a single-step heuristic but still suffers from high levels of error in inertial odometry.

[36] tracks a belief tensor containing all possible user states instead of individual particles to fuse map information with odometry data. States are propagated similar to a particle filter, and map information is used to downweight the probability of states which enter obstacles as described before. This method does not noticeably outperform a particle filter in their experiments and shares the problem of sensitivity to odometry noise of particle filter methods which use hand-defined map heuristics.

2.4 RSSI + Odometry + Map information

Multi-modal sensor fusion approaches that combine RSSI based methods with Odometry + Map information methods reduce the number of beacons required in a RSSI only method while using the global positioning information with the odometry-map combination to better constrain map information specially in cases of symmetrical or replicating map structures. [37] use a multi-step optimization technique using a CNN to combine the trajectory forecast from WiFi RSSI + odometry method with the map information.

Chapter 3

Approaches

We aim to develop an indoor navigation system that uses the different sensors available on a commodity smartphone while being unobtrusive and privacy preserving. To achieve this goal we started by developing a Bluetooth Low Energy (BLE) RSSI based localization module and solve the problem of BLE generalizing to multiple hardware from different manufacturers with very different signal characteristics. We then combine the BLE module with the state-of-the-art data driven inertial odometry method using a Particle Filter (PF) to reduce the number of beacons required to have high levels of accuracy. The BLE localization module provides global positioning and drift correction while the inertial module provides high frequency relative positioning and smooth motion. To further reduce the beacon requirement and improve accuracy by reducing the effect of drift, we encode floorplan (map) information using CNNs and use that as the transition model of the Particle Filter (PF). We deploy this localization system as mobile apps to ensure the efficiency and real-time capabilities of the system. We built iOS and Android apps that leverage the full array of sensors and perform real-time on-device localization thereby making it possible to have offline turn-by-turn navigation in real-time and with all the processing happening on-device while also having a reasonable battery consumption.

3.1 BLE Localization

We use RSSI based localization to provide accurate global positioning. Given a dense network of beacons, RSSI based methods are able to triangulate the user's location to decent levels of accuracy. WiFi RSSI scanning is not available to third party developers on the iOS platform for privacy concerns. To that extent, we decided to use Bluetooth Low Energy (BLE) beacons instead.

We use the iBeacon protocol developed by Apple for the BLE beacons. This allows easy scanning on both iOS and Android. We developed the BLE data collection apps on both iOS and Android to have a similar data collection and processing pipeline on both platforms.

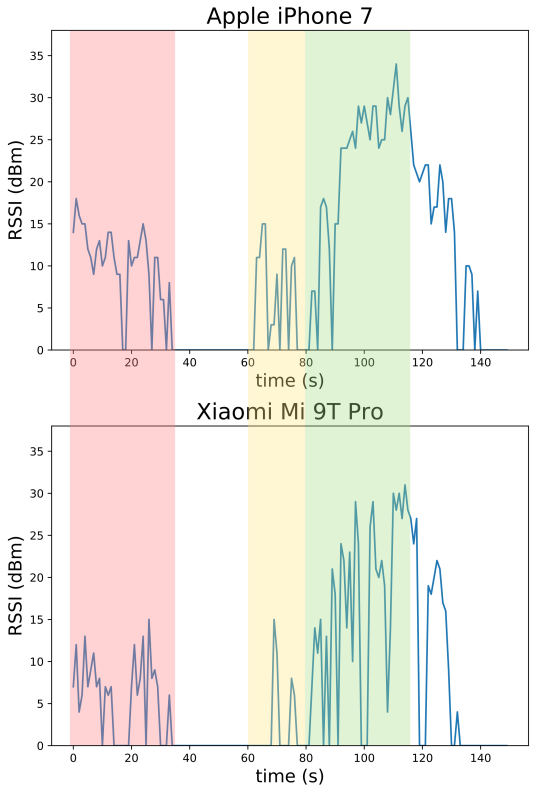


Figure 3.1: BLE signal comparison on different phones

In spite of our efforts to make the data collection process on both platforms as close to each other as possible, we still faced many challenges. As shown in figure

3.1 the BLE signal characteristics are not just different between iOS and Android but even vary drastically between phones from different manufacturers all running Android. To solve this problem, we decided to get a wide range of phones and train a self-supervised network to be able to model the noise introduced by different phones – even the phones the network has never seen at train time.

3.1.1 Dataset

Model	Price	OS	Processor
Apple iPhone XR	$600	iOS 13	A12 Bionic
Apple iPhone 8	$450	iOS 13	A11 Bionic
Apple iPhone 7	$190	iOS 13	A10 Fusion
Samsung Galaxy S10	$700	Android 10	Snapdragon SM8150 SD-855
Samsung Galaxy Note9	$600	Android 9	Snapdragon SDM845 SD-845
Samsung Galaxy A50	$300	Android 9	Exynos 9610
Google Pixel 4XL	$900	Android 10	Snapdragon SM8150 SD-855
Google Pixel 3XL	$550	Android 10	Snapdragon SDM845 SD-845
Google Pixel 3aXL	$400	Android 10	Snapdragon SDM670 SD-670
Xiaomi Mi 9	$435	Android 9	Snapdragon SM8150 SD-855
Xiaomi Mi 9T Pro	$360	Android 9	Snapdragon SM8150 SD-855
Xiaomi Redmi Note 8	$180	Android 9	Mediatek Helio G90T
Huawei Mate 20 Pro	$550	Android 9	HiSilicon Kirin 980
Huawei Honor 20 Pro	$340	Android 9	HiSilicon Kirin 980
Huawei Honor View 20	$290	Android 9	HiSilicon Kirin 980

Table 3.1: Smartphone models used as BLE Receivers.

To the best of our knowledge, [38] is the only publicly available BLE RSSI dataset. The dataset ([38]) was collected using iPhone6S, receiving BLE RSSI from 13 different iBeacons deployed in the Waldo Library of Western Michigan University. The dataset contains two sub-datasets: a labeled dataset (1420 instances) and an unlabeled dataset (5191 instances). As we try to analyze how varying phone types, change RSSI signals, the data cannot be used for our analysis.

We collected a large scale BLE dataset with about 54k RSSI fingerprint samples along with fine-grained location information, for 15 smartphone models. The setup was deployed across a university floor spanning 2000 sq. m. in area. The dataset

comprises of RSSI data ranging from BLE beacons for fifteen different phones from 5 major brands which include Apple, Samsung, Pixel, Huawei, and Xiaomi.

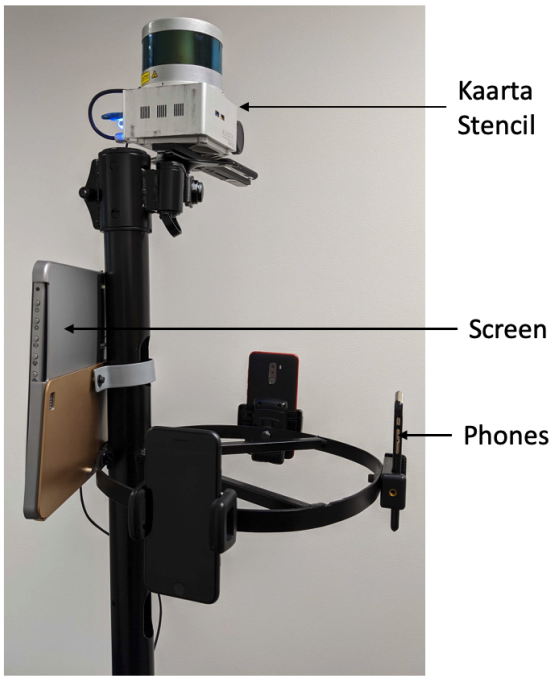


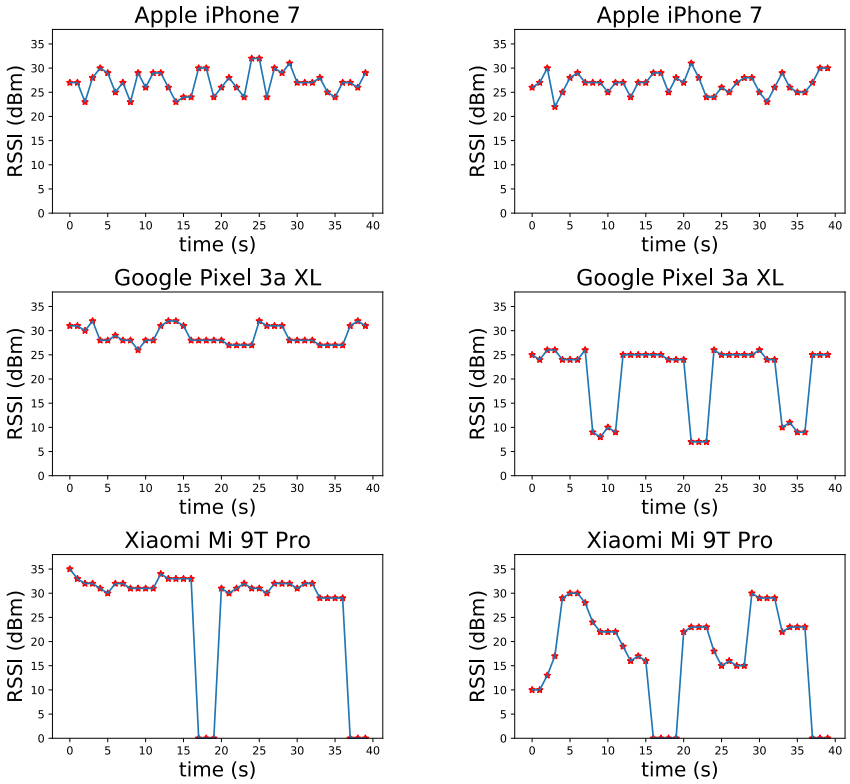
Figure 3.2: Data Collection Setup

We designed a rig, as shown in 3.2, which can hold three phones and a Kaarta Stencil, running LiDAR based localization algorithms, on top to get precise ground truth locations. Study in [39], suggests BLE data changes significantly with orientations, so having three facing directions helps us in augmenting the BLE data substantially at every single location. About 50 BLE beacons were installed all across the first floor of a university, as shown in 3.2.

Typically fingerprinting is performed standing at specific reference locations, collecting the RSSI and then using the BLE data at these reference locations to localize. However, as the data collection procedure is static, we cannot expect the methods to work very well when the robot/person is moving. Stencil by Kaarta is a stand-alone, lightweight and low-cost system delivering the integrated power of mapping and real-time position estimation with a precision of ±30mm. The device is

based on the scientific work [40] and depends on LiDAR, vision and IMU data for localization. The system uses a Velodyne VLP-16 connected to a low-cost MEMS IMU and a processing computer running Robotic Operating System (ROS) for real-time six DoF mapping and localization. A 10-Hz scan frequency is used for the data capture [41]. For robust estimates, we use the multi-modal localization engine, which gives us position estimates using data captured via both camera and LiDAR.

3.1.2 BLE RSSI Analysis



(a) w/o Interference (b) w Interference

Figure 3.3: RSSI vs. Time plots for the two experimental setups show that having multiple beacons in the environment has little impact on the nature of the signal from iPhone, some impact on the Pixel phone, but major impact in case of the Xiaomi phone.

Env. →	No Interference			W/ Interference		
Phone↓	**Mean**	**Var**	**SnR**	**Mean**	**Var**	**SnR**
iPhone 7	27.61	7.18	3.83	26.80	4.07	6.59
iPhone 8	30.20	1.73	17.49	27.52	8.25	3.34
iPhone XR	29.17	2.26	12.87	27.93	0.37	76.47
Galaxy S10	33.04	8.09	4.09	20.87	14.77	1.41
Galaxy Note 9	30.30	2.86	10.58	22.35	40.92	0.55
Galaxy A50	30.83	17.93	1.72	26.80	19.03	1.41
Pixel 4 XL	24.46	11.47	2.13	11.83	41.93	0.28
Pixel 3 XL	30.35	0.49	62.22	16.41	42.72	0.38
Pixel 3a XL	28.98	3.20	9.07	20.61	50.37	0.41
Mi 9	27.39	45.90	0.60	19.11	41.77	0.46
Mi 9T Pro	30.39	55.79	0.54	19.12	62.34	0.31
Redmi Note 8	32.33	4.35	7.43	22.50	43.95	0.51
Mate 20 Pro	34.67	1.39	24.88	26.33	21.87	1.20
Honor 20 Pro	28.04	11.91	2.35	21.07	35.63	0.59
Honor View 20	26.09	3.99	6.53	25.17	7.06	3.57

Table 3.2: Per phone signal statistics of one beacon in absence (w/o interference) and in presence (w/ interference) of other beacons. RSSI distribution changes significantly across phones.

In an isolated environment, we consider one beacon and one smartphone at a time. The BLE beacon and smartphone are kept at a distance of 1m, and RSSI data is collected for 60 seconds for each phone sequentially. We calculate the mean and the variance of the RSSI signal for each phone using the signal strengths registered during the 1-minute interval (Table 3.2 and Figure 3.1) Although the mean RSSI being registered across different phones is approximately the same, the variance changes drastically across phones. For some phones, signal drops are observed.

We repeat a similar experimental setup, with the same beacon, but in the presence of multiple other beacons to act as sources of interference. According to [15], we know multi-path fading, and interference due to multiple beacons can lead to a significant change in RSSI behavior. We observe 3.2 in addition to the variance, in this case, the mean RSSI changes significantly across phones, contrary to what we observed in the no interference setting. The variance is also not directly related to what we observed

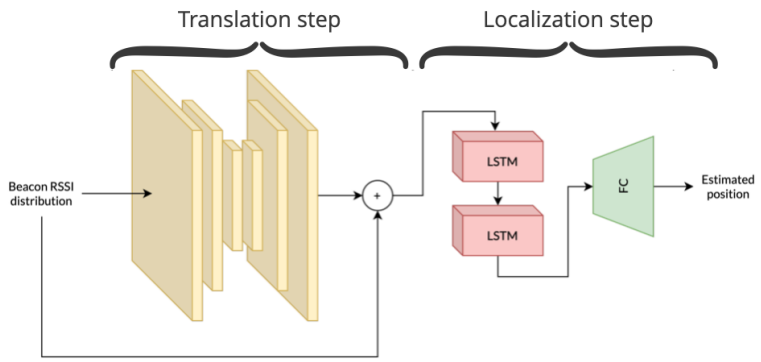


Figure 3.4: BLE Localization Model Architecture

in the absence of other sources of interference. For instance, Pixel3XL records a significant variance in this setup, and iPhone XR shows a remarkably stable reading, Figure (3.3). Empirically, we conclude, that the behaviour of RSSI distribution can change significantly across phones in the presence of other sources of interference.

3.1.3 Method

Consider an indoor environment with B beacons and P different smartphone models. Let the set of beacon indices be $\mathcal{B} = \{1..., B\}$, and the set of smartphone model indices be $\mathcal{P} = \{1, ..., P\}$.

At any time step t, smartphone $p \in \mathcal{P}$ is at location $\mathbf{x}_t^p \in \mathbb{R}^2$. The RSSI received by smartphone p from beacon $b \in \mathcal{B}$ at time t is $s_t^{p,b} \in \mathbb{R}_{\geq 0}$. The vector of RSSI received by smartphone p from all B beacons at time t is $\mathbf{s}_t^p = [s_t^{p,1} \;\; s_t^{p,2} \;\; ... \;\; s_t^{p,B}]^T \in \mathbb{R}_{\geq 0}^B$. For a history of H time steps, the matrix of RSSI received by smartphone p from all beacons at times $\{t-(H-1), t-(H-2), ..., t\}$ is $\mathbf{S}_t^p = [\mathbf{s}_{t-(H-1)}^p \;\; \mathbf{s}_{t-(H-2)}^p \;\; ... \;\; \mathbf{s}_t^p] \in \mathbb{R}_{\geq 0}^{B \times H}$ We would like to learn a mapping from the vector of RSSI received by a smartphone to the location of the smartphone, i.e., a mapping $L : \mathbb{R}^{|\mathcal{B}|} \to \mathbb{R}^2$

Consider the labeled dataset $\mathcal{D}_l = \{\langle \mathbf{S}_t^p, \mathbf{x}_t^p \rangle \mid p \in \mathcal{P},\ t \in \mathcal{T}_l\}$. Each element of $\mathcal{D}_l$ is a tuple, consisting of the matrix of RSSI received by a smartphone p, and its corresponding location at some time $t \in \mathcal{T}_l$.

Consider the unlabeled dataset $\mathcal{D}_u = \{\mathbf{S}_t^p \mid p \in \mathcal{P},\ t \in \mathcal{T}_u\}$. Each element of $\mathcal{D}_u$ is

the matrix of RSSI received by a smartphone p at some time $t \in \mathcal{T}_u$.

Let there be an unknown function $f : \mathbb{R}_{\geq 0}^{B \times H} \to \mathbb{R}^2$ mapping the matrix of RSSI received by any smartphone to the location of the smartphone at that time, i.e. $f(\mathbf{S}) = \mathbf{x}$. In the problem of indoor localization, our goal is to approximate this function.

We have access to limited labeled data from a subset of smartphone models $\mathcal{P}' \subset \mathcal{P}$, i.e., $\mathcal{D}_{\text{train}} = \mathcal{D}'_l$. Each model included in the subset $\mathcal{P}'$ belongs to a brand. For example, $\mathcal{P}'$ may include iPhone 7 and iPhone 8 from the Apple brand as well as Pixel 3 and Pixel 4 from the Google brand. In this example, $\mathcal{P}'$ does not include any models from Samsung, Xiaomi or Huawei brands.

If we learn a localization function using only the limited labeled data $\mathcal{D}'_l$, we will overfit to the smartphone brands included in $\mathcal{P}'$ since the training data is not representative of all smartphone brands. In order to avoid overfitting, we propose the use of a BLE signal translation function which maps RSSI measured by any smartphone to that of a smartphone brand which is known, *i.e.,* one which was part of the labeled training dataset.

Consider a brand-specific localization function $h : \mathbb{R}_{\geq 0}^{B \times H} \to \mathbb{R}^2$, which maps the matrix of RSSI measured by smartphone models of a known brand to the smartphone's location. Let the set of smartphone models of that known brand be indicated by $\mathcal{J} \subset \mathcal{P}'$.

Now consider a BLE signal translation function $g : \mathbb{R}_{\geq 0}^{B \times H} \to \mathbb{R}_{\geq 0}^{B \times H}$, which takes as input the matrix of RSSI received by any smartphone model $p \in \mathcal{P}$ and transforms it to a corresponding matrix of RSSI measured by a model $p' \in \mathcal{J}$ from the known smartphone brand, at the same location. Then, $f(\mathbf{S}) = h(g(\mathbf{S}))$ is the true localization function, where $\mathbf{S}$ is the matrix of RSSI received by a smartphone model $p \in \mathcal{P}$. Our goal is to learn an estimate of the localization function $\hat{f} = \hat{h}(\hat{g})$.

Signal Translation Network (`TransNet`): Our signal translation function $\hat{g}$ transforms the matrix of RSSI measured by a phone $p \in \mathcal{P}$, to the matrix of RSSI measured by a model of a known brand in the same location. Intuitively, we expect the difference in RSSI measured by different phones at the same location to be bounded. Therefore,

we constrain $\hat{g}$ to output a correction to its input. Formally, our signal translation function is a neural network of the form $\hat{g}(\mathbf{S}) = \text{ReLU}(\hat{r}(\mathbf{S}) + \mathbf{S})$. We call this network `TransNet`. The correction $\hat{r}(\mathbf{S})$ is a 6-layered 1-D convolutional autoencoder, composed of 3 encoder layers and 3 decoder layers with ReLU activations.

Localization Network (`LocNet`): The fully supervised localization network $\hat{f}$ is modeled as a 2 layer LSTM followed by two fully connected layers. We call the localization network, `LocNet`. At time t, it takes as input $\mathbf{S}_t^p$, i.e., the beacon readings received by smartphone p in the last $H = 5$ seconds, and outputs the location $\mathbf{x}_t^p$. During training, `LocNet` learns to correctly interpret temporal beacon information, e.g.: how fast signals are changing, patterns of oscillation in signals, etc., and maps it to a location $\mathbf{x}$.

This is the standard empirical risk minimization objective for supervised learning, where we learn the minimizer $\hat{f}$ over the training data for all phone model $\mathcal{P}$.

We define a new loss function with the following components:

1. **Localization loss:** $\mathcal{L}_{\text{loc}} = \left|\left|\hat{h}(\hat{g}(\mathbf{S}_t)) - \mathbf{x}_t\right|\right|_2^2$. This quantifies deviation of estimates of our localization function from ground truth label $\mathbf{x}$ and is identical to the loss function in scenario 1.
2. **Position smoothness loss:** This quantifies the distance between consecutive location predictions, $\mathcal{L}_{\text{ps}} = \left|\left|\hat{h}(\hat{g}(\mathbf{S}_t)) - \hat{h}(\hat{g}(\mathbf{S}_{t-1}))\right|\right|_2^2$. This term acts as a regularization term to ensure that estimated motion is smooth.
3. **Statistic similarity loss (SSL):** Intuitively, we know that beacons that are near each other will often be 'seen' together by the smartphone Bluetooth receiver. This means that beacon RSSI measurements are correlated and that the outputs of the translation function g should also be correlated. Formally, consider the subset of models $\mathcal{J} \subset \mathcal{P}'$ of the known brand . Labeled data from models of this brand is given by $\mathcal{D}'_J = \{\langle \mathbf{s}_t^p, \mathbf{x}_t^p \rangle \mid p \in \mathcal{J},\ t \in \mathcal{T}_l\}$. We use the brand-specific data $\mathcal{D}'_J$ to infer brand-specific correlation statistics.

   Now, consider a matrix $\mathbf{M} \in \mathbb{R}_{\geq 0}^{B \times B}$ of statistics derived from data $\mathcal{D}'_J$. Each

entry of $\mathbf{M}$ is described as:

$$\mathbf{M}_{ij} = \mathbb{E}_{\langle \mathbf{s},\mathbf{x} \rangle \sim \mathcal{D}'_j}[\mathbf{s}_j \mid \mathbf{s}_i > 0], \tag{3.1}$$

where $\mathbf{M}_{ij}$ is the expected RSSI from the j$^{\text{th}}$ beacon when the RSSI from the i$^{\text{th}}$ beacon is detected.

We use the inferred statistics to quantify the deviation of the output of $\hat{g}$ from the expected statistics of smartphone models $\in \mathcal{J}$. Our proposed statistic similarity loss (SSL) is defined by:

$$\mathcal{L}_{\text{ssl}} = \sum_{t=1}^{H} \sum_{i=1}^{B} \sum_{j=1}^{B} w_{it} d_{ijt} \tag{3.2}$$

where:

$$w_{it} = \exp\left(\frac{(\mathbf{S}_{it} - \mathbf{M}_{ii})}{\tau}\right), \quad \text{and} \tag{3.3}$$

$$d_{ijt} = \begin{cases} || \mathbf{M}_{ij} - \hat{g}(\mathbf{S})_{jt} ||_1 & \text{if} \quad \mathbf{M}_{ij} - \hat{g}(\mathbf{S})_{jt} < 0 \\ (\mathbf{M}_{ij} - \hat{g}(\mathbf{S})_{jt})^2 & \text{if} \quad \mathbf{M}_{ij} - \hat{g}(\mathbf{S})_{jt} \geq 0 \end{cases} \tag{3.4}$$

and τ is a hyperparameter. In our experiments, $\tau = 0.1$. Recall that the output of $\hat{g}$ is a matrix of translated RSSI and $\hat{g}(\mathbf{S})_{jt}$ denotes the entry corresponding to the j$^{\text{th}}$ beacon at time index t.

Our function $\hat{g}$ must map the matrix of RSSI received by any smartphone model to the corresponding RSSI received by a known smartphone model $\in \mathcal{J}$. Our proposed loss SSL quantifies the deviation of the output of $\hat{g}$ from the expected statistics of smartphone models $\in \mathcal{J}$. Specifically, $\mathcal{L}_{ssl}$ provides an exponential weighting (w_{it}) to any deviation (d_{ijt}) from the expected RSSI of the j$^{\text{th}}$ beacon when the i$^{\text{th}}$ beacon is detected, according to pairwise statistics $\mathbf{M}_{ij}$ of known smartphone models $\in \mathcal{J}$. When $\hat{g}$ is learned correctly the statistical deviation of the translated output signals should be low.

4. **Temporal smoothness loss:** One of the major problems we observed in

lower-end smartphone models is lost RSSI information where beacons randomly measure zero RSSI. In order to address this problem, we introduce a temporal smoothness loss that ensures that the output of the translation function $\hat{g}$ does not randomly drop RSSI,

$$\mathcal{L}_{\text{ts}} = \sum_{t=1}^{H-1} \sum_{i=1}^{B} || \, \hat{g}(\mathbf{S})_{i,t} - \hat{g}(\mathbf{S})_{i,t+1} \, ||_1. \tag{3.5}$$

$\hat{g}(\mathbf{S})_{i,t}$ is the entry corresponding to the i$^{\text{th}}$ beacon at time index t.

Optimization: We propose a modified optimization with weights w_{loc}, w_{ps}, w_{ssl}, w_{ts} on each term respectively:

$$\min_{\hat{h},\, \hat{g}} \mathbb{E}_{\langle \mathbf{S}, \mathbf{x} \rangle \sim \mathcal{D}_l'} \left[w_{\text{loc}} \mathcal{L}_{\text{loc}} + w_{\text{ps}} \mathcal{L}_{\text{ps}} + w_{\text{ssl}} \mathcal{L}_{\text{ssl}} + w_{\text{ts}} \mathcal{L}_{\text{ts}} \right] \tag{3.6}$$

By training $\hat{h}$ and $\hat{g}$ to minimize this objective, we expect generalization of localization performance to unseen smartphones.

3.1.4 Results

Scenario 1: Labeled Data for All Smartphone Models

In our first scenario, we have labeled data from all smartphone models, *i.e.,* $D_{\text{train}} = \mathcal{D}_l$. We train the localization network `LocNet`, using the Adam optimizer for 50 epochs, with a learning rate = 10^{-4} and batch size = 256 samples.

Results: We report mean and standard deviation of absolute localization error over test data in Table 3.3. Note, we do not have any numbers in bold here, since we do not compare our proposed method with any other method in this scenario. In this ideal scenario, since we have labeled data for all phones, performance on all phones is very good, with an average AE of **1.37 m** over all test data. Despite the high amounts of receiver failure for smartphone models from Huawei and Xiaomi

Scenario →	**Scenario 1**		**Scenario 2**				**Scenario 3**			
	Proposed		**Baseline 1**		**Baseline 2**		**Proposed**		**Proposed** (Huawei)		**Proposed** (Xiaomi)	
Test Phone	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
iPhone 7	0.95	0.65	0.99	0.83	0.92	0.83	**0.91**	**0.75**	0.95	0.88	**0.94**	**0.82**
iPhone 8	1.09	0.73	1.13	0.95	1.16	0.99	**1.06**	**0.76**	1.11	**0.83**	**1.07**	**0.83**
iPhone XR	0.92	0.60	1.05	0.83	0.95	0.75	**0.84**	**0.60**	0.89	0.80	**0.86**	**0.63**
Mate 20 Pro	1.96	1.61	4.19	4.25	2.62	2.45	**1.76**	**1.61**	**1.63**	1.32	**1.63**	**1.31**
Honor 20 Pro	1.92	1.46	3.14	2.19	2.36	1.77	**1.95**	**1.42**	1.99	**1.33**	**1.92**	1.34
View 20	1.74	1.40	3.15	3.31	2.20	2.04	**1.64**	**1.59**	1.66	1.32	**1.64**	**1.29**
Mi 9	1.53	1.20	3.82	5.48	2.39	3.30	**1.69**	**1.57**	1.65	1.31	**1.58**	**1.11**
Mi 9T Pro	1.77	1.14	3.78	4.98	2.71	3.35	**1.78**	**1.41**	1.84	1.45	**1.68**	**1.20**
Redmi Note 8	1.50	1.17	2.25	2.01	1.76	1.57	**1.62**	**1.16**	**1.61**	1.09	**1.61**	**1.06**
Pixel 3a XL	1.05	0.83	1.51	1.30	1.25	1.12	**1.24**	**1.07**	1.22	1.07	**1.21**	**1.06**
Pixel 3XL	1.21	0.96	2.04	1.56	1.54	1.29	**1.22**	**0.95**	**1.17**	0.96	1.18	**0.94**
Pixel 4XL	1.21	0.83	2.13	1.64	1.53	1.28	**1.36**	**1.01**	**1.39**	**1.07**	1.40	1.08
Samsung S10	1.44	1.15	1.80	1.41	1.58	1.35	**1.31**	**1.05**	1.30	1.02	**1.23**	**0.93**
Samsung A50	1.49	0.98	2.11	1.68	1.62	1.34	**1.42**	**1.30**	**1.36**	**1.22**	1.42	1.26
Samsung Note9	1.14	0.87	1.67	1.23	1.27	0.98	**1.13**	**0.82**	**1.15**	**0.81**	**1.15**	0.82
Overall Average	1.37	1.11	2.23	2.76	1.67	1.83	**1.37**	**1.20**	1.37	1.15	**1.34**	**1.09**

Table 3.3: Mean and std. deviation of absolute localization error for all methods and scenarios. All numbers are in meters(m). Numbers in bold indicate best performance within each scenario. Numbers in color indicate best performance overall.

brands, AE for each of these models is **less than 2 m**. However, as noted before, it is unrealistic to assume that we can performing fingerprinting for all smartphone models.

Scenario 2: Limited Labelled data

In our second scenario, we have only limited labeled data $\mathcal{D}_{\text{train}} = \mathcal{D}'_l$ from a subset of smartphone models $\mathcal{P}' \subset \mathcal{P}$. In our experiments with this scenario, $\mathcal{P}'$ includes models of the two most popular smartphone brands [1] - Apple and Samsung, *i.e.,* {iPhone 7, iPhone 8, iPhone XR, Samsung S10, Samsung A50, Samsung Note 9}. Our signal translation function $\hat{g}$ learns to transform RSSI measured by any smartphone model to that measured by the Apple brand, *i.e.,* $\mathcal{J} \subset \mathcal{P}'$ includes smartphone models

[1] as reported by the IDC - Worldwide Quarterly Mobile Phone Tracker

Stat. ↓	Scen. 1	Scen. 2			Scen. 3	
	Prop.	**Base. 1**	**Base. 2**	**Prop.**	**Prop.** (Hua.)	**Prop.** (Xia.)
Mean	1.37	2.23	1.67	1.37	1.37	**1.34**
Std. Dev.	1.11	2.76	1.83	1.20	1.15	**1.09**
Median	1.07	1.47	1.15	**1.02**	1.06	1.04
90 %ile	2.68	4.55	3.49	2.76	2.76	**2.66**
Max.	12.28	37.58	29.67	19.72	13.19	**11.82**

Table 3.4: Various statistics of absolute localization error over all test data for all methods and scenarios. All numbers are in meters (m). Numbers in bold indicate best performance over all methods.

Method →	KNN Reg		Bayesian		KRR+KF		DL-RNN		Proposed(Sup)		Proposed (Semi)	
Test on ↓	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
iPhone	1.1917	0.9298	1.5424	1.2267	1.7575	1.0770	1.0205	0.7712	**0.9409**	**0.7163**	0.9572	0.7741
Huawei	6.3579	9.595	4.8951	6.3216	3.0021	2.3075	2.2437	2.2014	1.7900	1.5515	**1.7370**	**1.3260**
Xiaomi	4.1781	7.2262	3.5959	4.2674	2.7569	1.9125	2.4360	3.0378	1.7014	1.3918	**1.6284**	**1.1280**
Pixel	1.9082	1.5561	2.1457	1.7167	1.8955	1.1816	1.4347	1.1638	1.2769	**1.0183**	**1.2687**	1.0341
Samsung	2.0139	2.5525	2.0843	2.3115	2.0767	1.2390	1.5282	1.1301	1.2898	1.0808	**1.2732**	**1.0310**
Average	3.0157	5.7190	2.7724	3.8090	2.2640	1.6644	1.6826	1.8650	1.3700	1.2080	**1.3462**	**1.0979**
Median	1.5230		1.7505		1.8884		1.2067		**1.0258**		1.0436	
90% ile	5.2514		5.5866		4.2660		3.2968		2.7689		**2.6601**	
Max	52.5572		48.3013		17.2977		31.9299		19.7231		**11.8294**	

Table 3.5: Evaluation results for different methods when trained using data from iPhone and Samsung phones

from the Apple brand {iPhone 7, iPhone8, iPhone XR}. Our localization function in this scenario is $\hat{f} = \hat{h}(\hat{g})$, where $\hat{h}$ is `LocNet` and $\hat{g}$ is `TransNet`. We train $\hat{f}$ using the Adam optimizer for 50 epochs, with a learning rate $= 10^{-4}$ and batch size $= 256$ samples. Appropriate weights for each loss term are obtained using validation data, which was a randomly chosen independent run from the training data for each phone. we use the validation data from only $\mathcal{D}'_l$

Baselines: We compare our method against the following baselines:

1. *Baseline 1*: Similar to the first scenario, we consider the situation where our localization function $\hat{f}$ is `LocNet` only. We limit the available labeled data

to that from smartphone models of the Apple brand only, *i.e.,* $\mathcal{P}'$ includes {iPhone7, iPhone8, iPhone XR}. We train `LocNet` using labeled data from the Apple brand. We include this baseline to verify our hypothesis that data augmentation alone is not enough to generalize well to unseen phones. We expect our proposed method to outperform this method due to the inclusion of labeled data from an additional smartphone brand.

2. *Baseline 2*: Again, we consider the situation where our localization function $\hat{f}$ is `LocNet` only. However, for a stronger baseline, we consider that we have access to the same data as our proposed method, i.e., $\mathcal{P}'$ includes models from the Apple and Samsung brands. We train `LocNet` using labeled data from Apple and Samsung brands. We include this baseline to verify our hypothesis that simply training `LocNet` using limited labeled data will lead $\hat{f}$ to overfit to the known phone brands. We expect our proposed method to outperform this method due to the inclusion of a signal translation function $\hat{g}$. This is because $\hat{g}$ explicitly learns to translate RSSI measured by a (possibly unknown) smartphone to that measured by a known brand, using our novel statistic similarity loss (SSL). This enables $\hat{f} = \hat{h}(\hat{g})$ to generalize better to unseen phones.

Results: From Table 3.3, we see that our proposed method significantly outperforms baselines in terms of both mean and standard deviation of AE over all test data, as well as over test data from individual smartphone models. Moreover, our proposed approach performs as well as the fully supervised approach from the first scenario, achieving an average AE of **1.37 m**, despite having access to only limited labeled data. Importantly, this demonstrates its ability to generalize well to unseen smartphone models.

While Baseline 2 performs significantly better than Baseline 1, it performs poorly on unseen smartphone brands Huawei and Xiaomi. By comparison, our proposed approach consistently gives mean AE of **less than 2 m** for models of these brands. Moreover, from Table 3.4, Baseline 2 shows a high maximum error of $\approx 30m$. This

supports our hypothesis that training with limited data leads `LocNet` to overfit to known smartphones. Our proposed approach avoids this problem by training an explicit signal translation function $\hat{g}$ using our novel SSL loss.

Baseline 1 performs very well on iPhone models, with mean and standard deviation of AE within 1 m. However, its performance is significantly worse on other smartphone brands, especially Huawei and Xiaomi. Moreover, from Table 3.4, the maximum and 90 percentile errors are the highest in the table. Baseline 1 overfits to a larger degree than Baseline 2, supporting our hypothesis that data augmentation alone is not enough to generalize well to unseen phones.

Scenario 3: Limited Labeled and Unlabeled Data

In our third scenario, we have limited labeled data $\mathcal{D}'_l$ from a subset of smartphone models $\mathcal{P}' \subset \mathcal{P}$. Additionally, we have unlabeled data $\mathcal{D}'_u$ from other smartphone models $\mathcal{Q}' \subset \{\mathcal{P} \setminus \mathcal{P}'\}$. Our training data is thus $\mathcal{D}_{\text{train}} = \mathcal{D}'_l \cup \mathcal{D}'_u$. Similar to the previous scenario, in our experiments with this scenario, $\mathcal{P}'$ includes models of Apple and Samsung brands. Our signal translation function $\hat{g}$ learns to transform RSSI measured by any smartphone model to that measured by the Apple brand. We consider $\mathcal{Q}'$ to include smartphone models of the Huawei brand or the Xiaomi brand. Our localization function in this scenario is $\hat{f} = \hat{h}(\hat{g})$, where $\hat{h}$ is `LocNet` and $\hat{g}$ is `TransNet`. In this scenario,

1. We train $\hat{f}$ using the Adam optimizer for 50 epochs, with a learning rate $= 10^{-4}$ and batch size $= 256$ samples. Appropriate weights for each loss term are obtained using validation data.
2. We then train $\hat{f}$ for the **semi-supervised** objective using the Adam optimizer for 20 epochs, with a learning rate $= 10^{-5}$ and batch size $= 256$ samples.

Results: From Table 3.3, we see that our proposed method trained in a semi-supervised fashion using unlabeled data from the Xiaomi brand leads to better test performance for Xiaomi and Huawei smartphones, when compared to our proposed method in scenario 2. Moreover, our proposed method **outperforms the fully**

supervised method in scenario 1, with a mean AE of **1.34 m** and standard deviation AE of **1.09 m** over all test data. Importantly, this result is achieved despite only having access to limited labeled data, demonstrating our proposed method's ability to generalize well to unseen phones. From Table 3.4, our proposed method leads to the best (or nearly the best) performance on all metrics over all the test data.

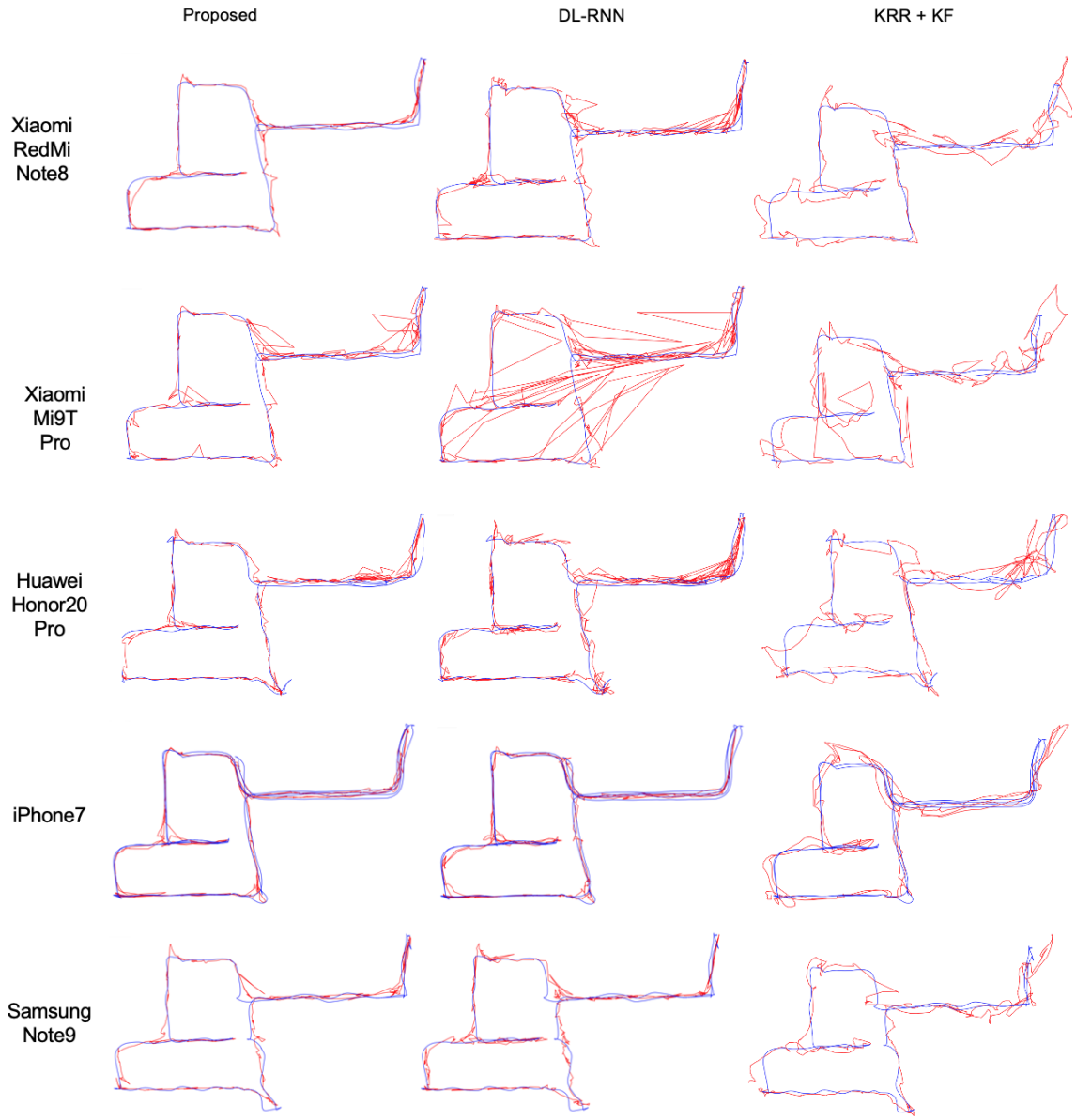


Figure 3.5: Visualization of Localization Performance comparing proposed approach to other approaches. Blue denotes ground-truth, red denotes predictions

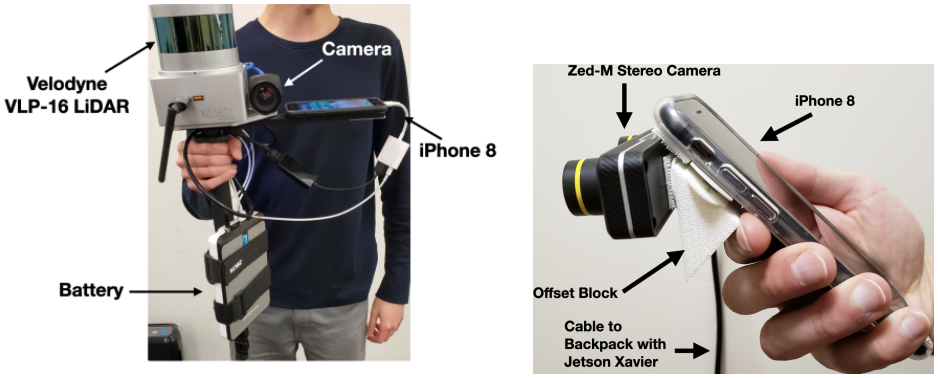
3.2 BLE + IMU + Map Prior

In this section, we aim to develop a system which consumes high drift odometry measurements and uses indoor occupancy map information to reduce drift in the final estimated user location. Map information is integrated via the sensor model of a particle filter, but prior methods of doing so are sensitive to high levels of error in inertial odometry. Instead, we learn a data-driven map prior from odometry and building maps which we hypothesize will indicate likely user locations with higher accuracy than traditional heuristic methods. We also combine the output of the BLE RSSI based localization model in the particle filter to get estimated location in the absolute world frame and not in the relative inertial frame.

3.2.1 Dataset

IDOL Dataset: The IDOL dataset [27] is an inertial odometry dataset for indoor pedestrian localization. It consists of 20 hours of 10 minute pedestrian trajectories collected by users carrying a smartphone in 3 buildings (Figure 3.6a). Smartphone IMU data and ground truth device pose is available at 100Hz. Odometry measurements are generated from IMU data by the IDOL method proposed in the same work. 2D occupancy maps are generated from the architectural plans available for the 3 buildings.

BLE + IMU Dataset: We collect a smaller dataset similar to the IDOL dataset but with the addition of Bluetooth low energy (BLE) beacon measurements, to compare against bluetooth-based localization. Two floors of a building like those used in the IDOL dataset are instrumented with BLE beacons emitting a unique identifier broadcast at 1Hz, which are recorded along with the broadcast signal strength and IMU data by a smartphone at 60Hz. Ground truth device pose and orientation are generated with a Stereolabs Zed Mini stereo camera configured as in Figure 3.6b.



(a) IDOL dataset (b) BLE + IMU dataset

Figure 3.6: Data collection setups for sensor fusion.

3.2.2 Method

Our learnable Map Prior Network (MPN) is shown in Figure 3.7. The network aims to generate the likelihood score that a given series of odometry measurements ends at a given location $\mathbf{x}$ in the map.

The likelihood score can be used directly as the particle filter weight for a particle at $\mathbf{x}$. We set up this network with two branches to separate map and odometry processing, whose deep encodings are later combined. This setup is useful because maps are generally static, so at runtime a specific indoor map only needs to be processed by the network once to generate its encoding. The stored map encoding can later be combined with the constantly changing odometry encoding, reducing computational burden. The inputs to our method are a 2D indoor occupancy map $\mathtt{M}$ and a short history of the agent's most recent odometry $\mathtt{O}$ from time $t-n$ to t. n is selected experimentally to be 5 seconds for human walking. We utilize a particle filter to combine the learned map prior with odometry measurements and BLE localization output. Particle filters have relatively low computational burden but are able to handle complex distributions of estimated location. A particle filter maintains p estimates of the user's state (location) as particles. At each timestep, each particle

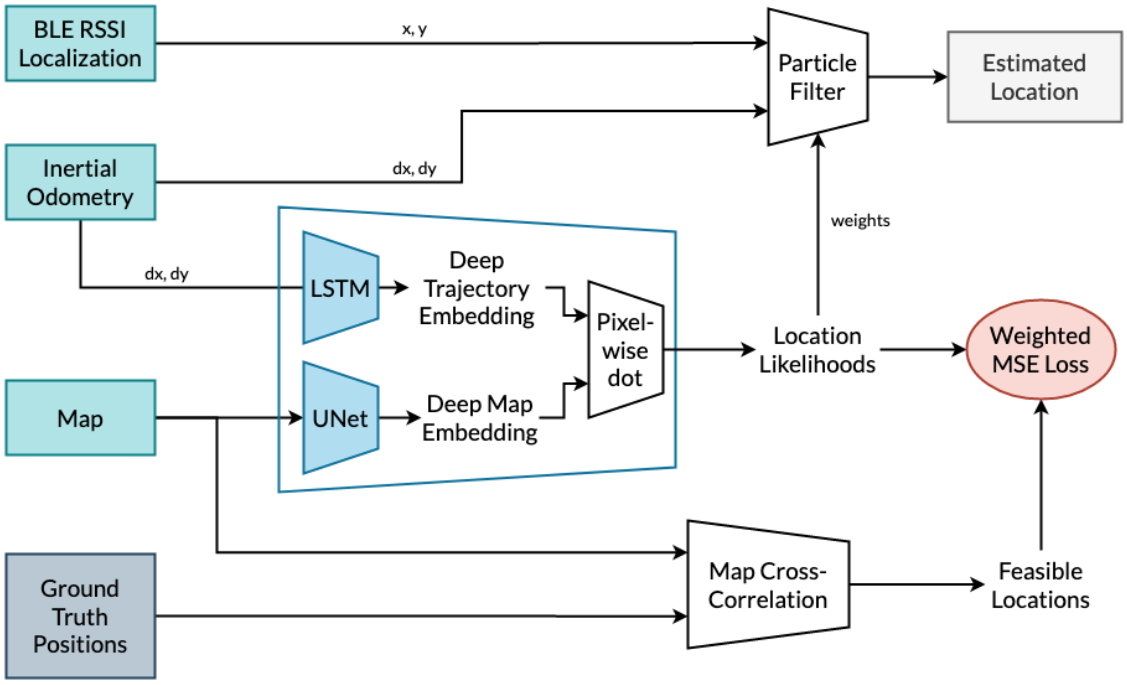


Figure 3.7: System Diagram: Our map prior network takes as input (1) occupancy map and (2) window of odometry measurements, and outputs a map of likelihood scores. Scores represent map locations that are likely given input odometry measurements. Scores used to weigh samples of a particle filter for tracking agent's trajectory.

has its stored state updated according to the motion model:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta\mathbf{x} + \eta_{\text{motion}} \tag{3.7}$$

where $\mathbf{x}_t$ is the user's state (x, y) at timestep t, and $\Delta\mathbf{x}$ is the odometry measurement between times t and $t+1$. 2D Gaussian noise $\eta_{\text{motion}} \sim \mathcal{N}(\mu, \Sigma)$ is added to each particle's motion to capture odometry uncertainty.

Our map network $f(\mathtt{M}, \Theta_f)$ is a small version of U-Net [42, 43] parameterized by weights Θ_f which has been modified to output a "Deep Map Tensor" of the same spatial dimensions as the input map, but with more channels. The number of channels $c = 32$ is chosen experimentally, balancing speed of computation with sufficient space to store relevant environmental details. The odometry network $g(\mathtt{O}, \Theta_g)$ is a two-layer Long-Short-Term Memory (LSTM) network parameterized by weights Θ_g operating

on the time dimension of the odometry history. The LSTM hidden state is of size c, and the last hidden state output is used as the "Deep Trajectory Vector". A dot product is taken between the Deep Map Tensor and the Deep Trajectory Vector at each pixel location $\mathbf{x}$, giving the likelihood score of $\mathbf{x}$ given $\mathtt{O}$ and $\mathtt{M}$:

$$\mathcal{S}(\mathbf{x}|\mathtt{M},\mathtt{O}) = f(\mathtt{M},\Theta_f)_{\mathbf{x}} \cdot g(\mathtt{O},\Theta_g) \tag{3.8}$$

Particle weights for the particle filter are computed using the value of the location likelihood score heatmap at the particle's location from the last n seconds of odometry and the localization estimate from the BLE RSSI based localization model following –

$$w^i = \mathcal{N}(\|\mathbf{x}^{\mathrm{i}} - \mathrm{z}^{\mathrm{BLE}}\|, 0, 3) \tag{3.9}$$

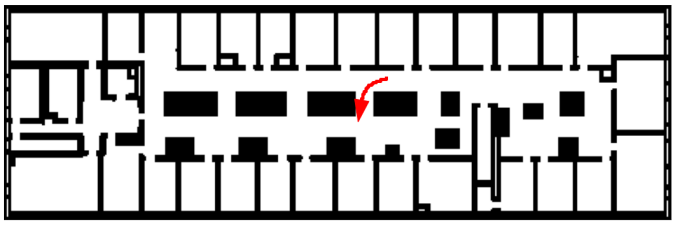
$$\mathbf{x}_{\mathrm{t+1}} = \sum_i w^i x^i_{t+1} \tag{3.10}$$

Low variance resampling of the particle cloud is used due to its stability and consistent particle space coverage [44]. Resampling copies particles from the original cloud with probability proportional to their weight. The particle closest to the median (as opposed to the mean) of the new particle cloud is used as the estimated location of the agent. This guarantees the estimated location is feasible and not inside an obstacle even with a multi-modal particle distribution.

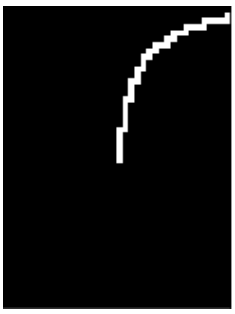
We also implement a re-initialization method for situations where the particle filter has diverged from a reasonable estimate. We reinitialize by sampling particles randomly within a radius r_{reinit} of the last estimated location when more than s_{reinit} of particles have passed through an obstacle. We experimentally set $r_{\mathrm{reinit}} = 5$ meters and $s_{\mathrm{reinit}} = 90\%$.

Training: To train our model, we window ground truth walking trajectory data into windows of n timestamps, and subtract the initial position in each window to make this data look like odometry data. We additionally augment the training data and add a multiplicative velocity bias drawn from $\mathcal{N}(1, 0.5)$ pixel/second and a random noise drawn from $\mathcal{N}(0, 0.25)$ pixels to each input sample to better mimic noisy odometry

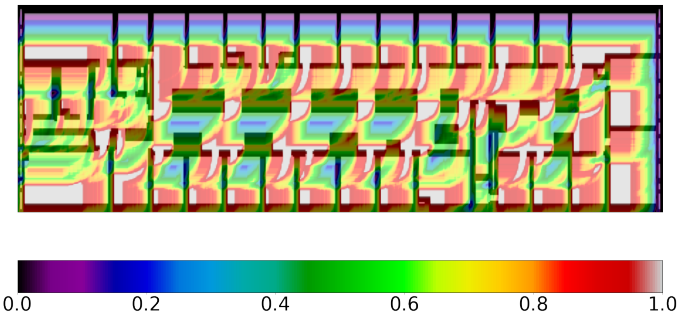
data. These windows are used as odometry branch training input. The map branch is given small segments of the relevant map at train time to avoid overfitting to the most commonly feasible areas of the full map.



(a) Example input ground truth trajectory window (in red).



(b) Cross correlation kernel.



(c) Resulting training target obtained by cross-correlating the kernel with the input map. Higher values are trajectory end locations that are more feasible.

Figure 3.8: Ground truth generation process for training our map prior network.

We create a target $\mathtt{T}$ (Figure 3.8c) defining which areas of the map are feasible end-locations for a given ground-truth trajectory window. We train our network to output $\mathtt{T}$, even when the odometry it is given as input is noisy or drifting. A 2D kernel like in Figure 3.8b is created from the ground truth trajectory segment and is cross-correlated with the map to form $\bar{\mathtt{T}}$ whose values depend on the input trajectory's overlap with obstacles. $\mathtt{T} = 10^{-6} * e^{14*\bar{\mathtt{T}}}$ is computed to heavily upweight areas where the trajectory window passes only through free space, *i.e.* is physically feasible. We train using Mean Squared Error (MSE) loss: $\mathcal{L} = \text{MSE}(\mathtt{T}, \mathcal{S}(\boldsymbol{x}|\mathtt{M}, \mathtt{O}))$. In training our

method we noticed that smaller feasible regions of T were not being predicted well. We weight feasible regions by the inverse of their pixel area in our loss to improve small area prediction.

3.2.3 Results

We evaluate our method in five experiment settings. (1) **BLE only (dense)** where we consider a dense network of beacons and use the state-of-the-art BLE only localization model. (2) **BLE only (sparse)** where we drastically reduce the number of beacons and use the same BLE localization model trained with the sparse set of beacons and then evaluated. (3) **IMU only** where we use the state-of-the-art inertial odometry models and show the ability of our learned map prior to reduce drift compared to IMU only odometry and better than other heuristic map priors. (4) **BLE (sparse) + IMU** where we show that a particle filter combining IMU and a sparse network of BLE beacons can improve the BLE only model with a sparse network of beacons and bring it close to the accuracy of BLE model with a dense network of beacons. (5) **BLE + IMU + Map prior** where we show the ability of the map prior network to further improve accuracy of the BLE + IMU fusion method by reducing drift from inertial methods and uncertainty from high variance in BLE RSSI signals.

Metrics: We evaluate localization performance using two main metrics: Absolute Trajectory Error (ATE) and End Error (EE). ATE is the root mean square error between points in the estimated and ground truth trajectories. The error at timestamp i, $e_i = \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2$. ATE measures global trajectory consistency and tends to increase with time due to drift. EE is the distance between the final estimated and final ground truth positions in a trajectory, i.e. $e = \|\mathbf{x}_T - \hat{\mathbf{x}}_T\|_2$. EE is a measure of the total accumulated drift in the sequence.

Training/Testing: We use Adam optimizer for training with a learning rate of 0.01. A batch size of 32 is used and the network takes between 50 and 100 full passes through the data (epochs) to converge, depending on the map.

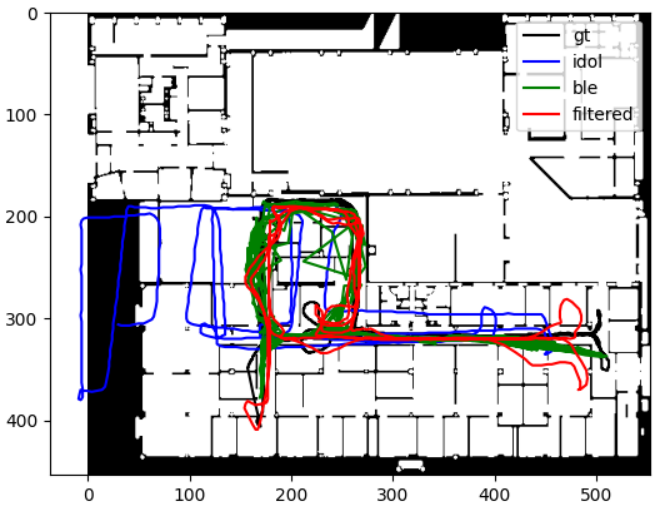
At test time the particle filter is initialized with 1000 normally distributed particles ($\sigma = 0.01$m) around the true starting location obtained from the BLE localization model. The filter runs at 1 Hz. The motion model uses a white Gaussian noise with $\Sigma = 0.1\mathcal{I}_2$m. A benefit of our formulation is the decomposition of trajectory and map processing. Once the network is trained, the map only needs to be passed through the network once to obtain a Deep Map Tensor. This can be reused for all trajectories in the same map. At test time, only the trajectory LSTM needs to process data. Our particle filter runs at 4x real-time speed on a desktop processor, suggesting that our method would be feasible for smartphone or low-compute platforms.

	Bldg. 1		**Bldg 2.**		**Bldg 3.**	
Method ↓	**ATE**	**EE**	**ATE**	**EE**	**ATE**	**EE**
PDR	24.28	24.96	12.66	13.80	21.86	22.18
IDOL	5.65	8.51	6.62	10.61	8.33	9.71
CRF	4.66	6.58	7.56	10.54	8.82	**7.38**
Heuristic PF	3.11	3.42	11.37	15.36	6.71	9.67
Learned (Ours)	**2.87**	**1.60**	**2.51**	**6.08**	**5.66**	9.99

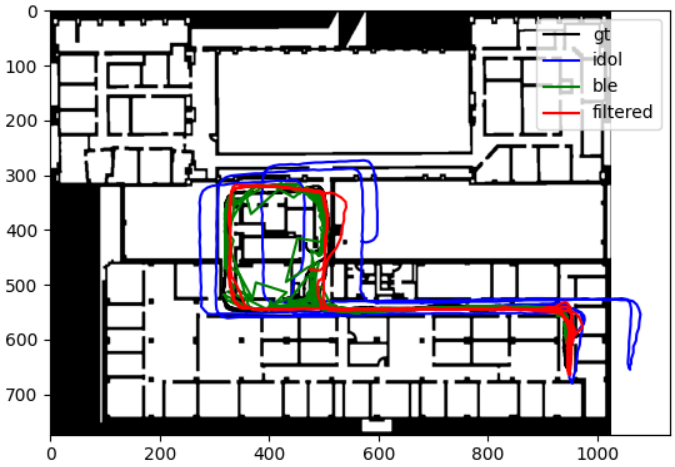
Table 3.6: Quantitative results of our approach compared to baselines on the IDOL[27] dataset. We report mean trajectory errors over all sequences per building.

	Bldg. 1		**Bldg 2.**	
Exp. ↓	**ATE**	**EE**	**ATE**	**EE**
Dense BLE (#beacons = 22)	1.92	1.43	1.88	1.37
Sparse BLE (#beacons = 4)	3.83	4.23	3.18	3.92
Inertial Odometry	9.92	10.23	4.48	7.43
Sparse BLE + Inertial	2.63	3.13	2.41	2.98
Inertial + Map	2.38	3.23	2.32	3.12
Sparse BLE + Inertial + Map	1.99	1.62	2.04	1.48

Table 3.7: Quantitative results of different sensor fusion approaches on the BLE + IMU dataset we collected.



(a) Building 1



(b) Building 2

Figure 3.9: Qualitative comparison of different sensor fusion approaches on BLE + IMU dataset.

Chapter 4

System Design

We deploy the localization algorithms as mobile apps on the two most popular mobile platforms – iOS and Android. We use PyTorch mobile to optimize the deep learning models and deploy them to run on-device for both the mobile platforms. The app architecture has been designed to be modular and easily upgradable to the state-of-the-art localization models.

4.1 App design

The app has been designed to have a clean and modern design that can work well on both iOS and Android. We considered designing for different color-blindness. We ended up designing a black and white UI for maximum contrast and legibility irrespective of the nature of color-blindness. We designed both a light and dark mode that looks native to both the platforms. UI consistency between the two platforms was important while maintaining the native feel of the specific platform. To that extent, we tried to keep the UI consistent only changing elements for the two platforms when absolutely necessary to give it a native feel. We leverage an open-source map rendering library – MapLibre[1] – to render the maps using hardware accelerated OpenGL calls.

[1]https://maplibre.org/

4.1.1 iOS

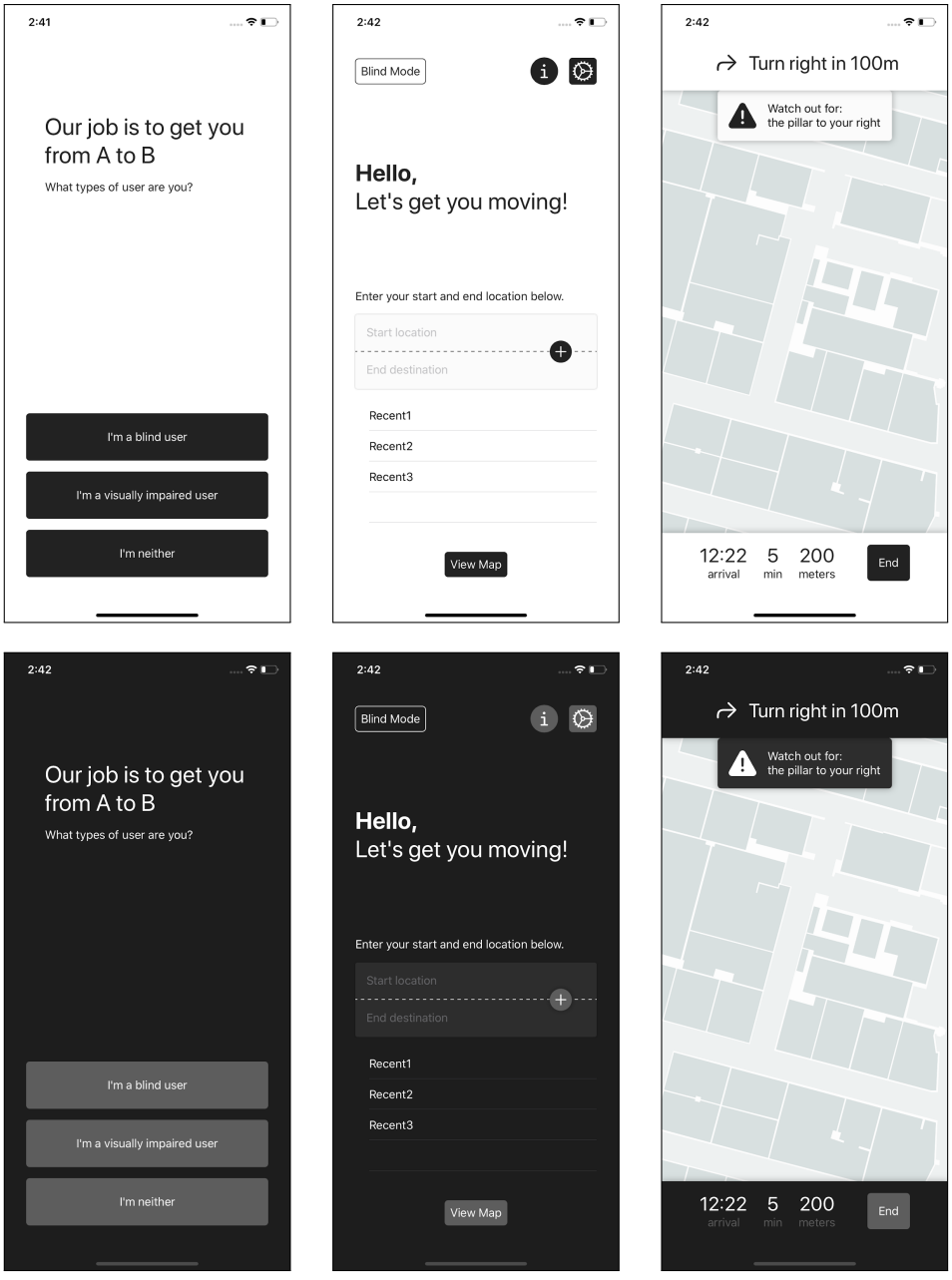


Figure 4.1: iOS app UI

4.1.2 Android

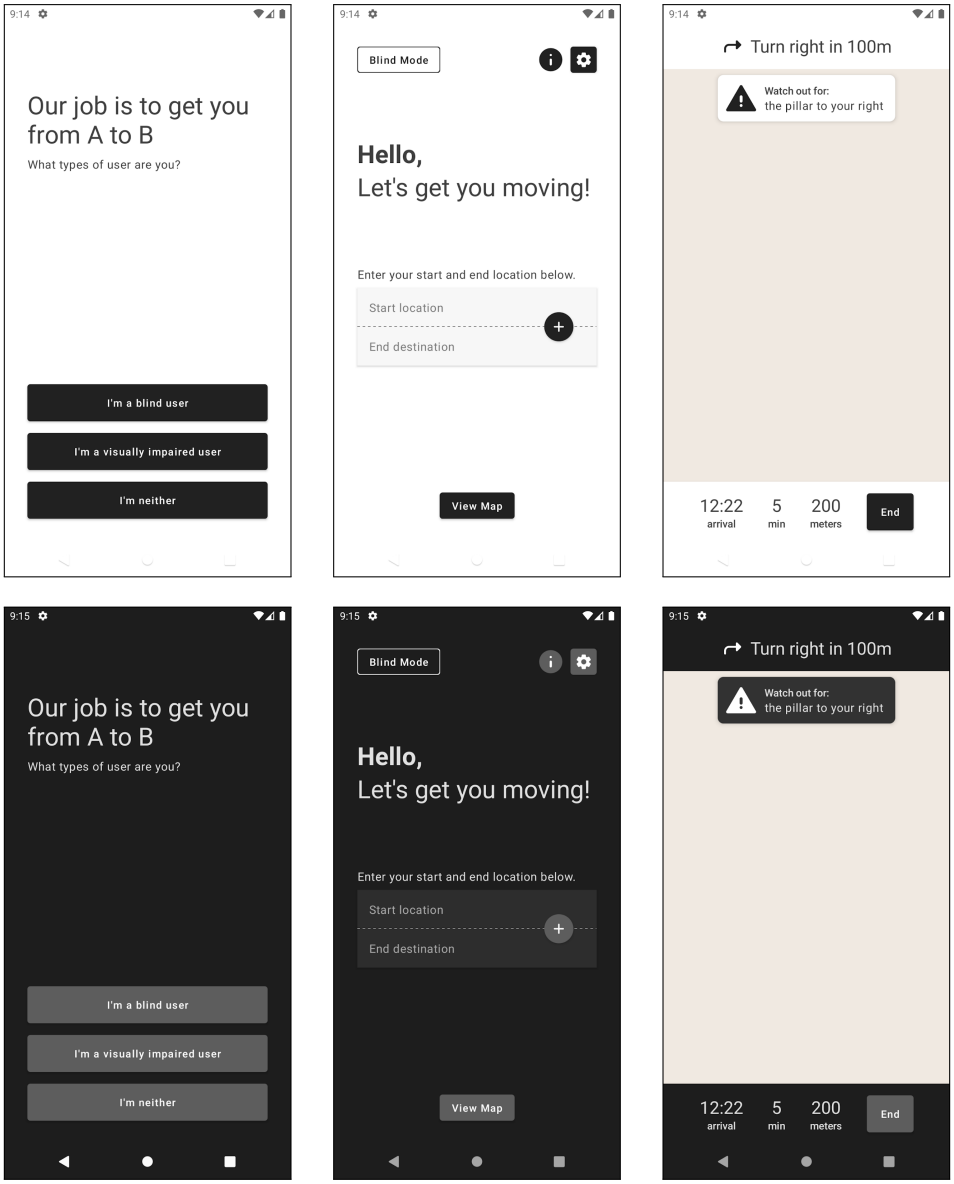


Figure 4.2: Android app UI

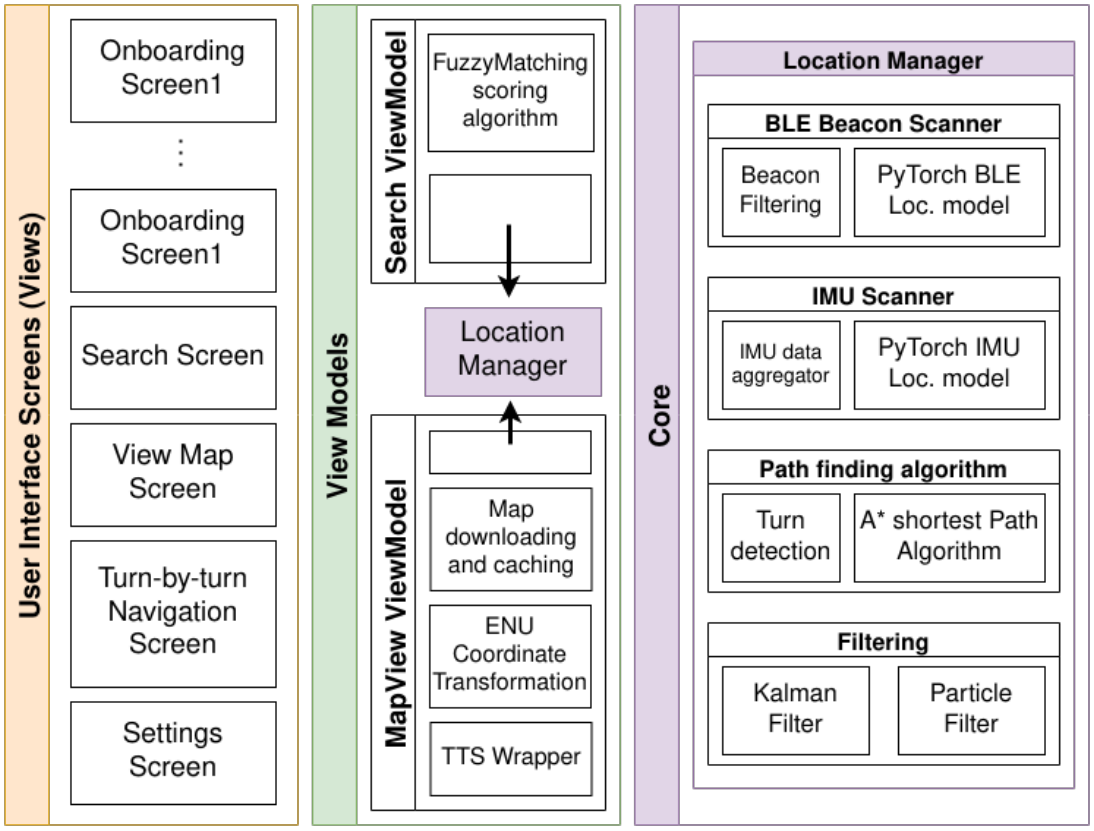


Figure 4.3: App Architecture for both iOS and Android

4.2 App Architecture

The architecture of the app has been designed with consistency and modularity in mind. The different modules have been designed so that the core can be changed easily, i.e. the models can be updated or different modules can be added to it without affecting the rest of the app. This lets us have a common interface Localization Manager that can have different sensor models and sensor fusion modules while keeping the rest of the app opaque to the implementation details. The architecture is also designed so that it is consistent in both iOS and Android thereby helping with maintaining two versions of the app and allowing easy replication of changes.

4.3 Privacy and Performance

Privacy and performance have been the key considerations from the very beginning. While a lot of other methods for indoor localization and navigation use cameras and visual inertial odometry to have highly accurate estimates, we decided to not use cameras for privacy and performance reasons. Having an always on camera recording is good neither for privacy nor performance as it will lead to high computation requirements and battery drain. Our localization module runs on-device in real-time on an iPhone 8 which was released 5 years ago. We measured the inference time on an iPhone 8 to be $\approx 20ms$ and $\approx 38ms$ for the BLE and IMU localization models respectively. We run this inference at 1Hz rate and recorded the battery drain to be 9-11% per hour of use depending on the battery level of the phone. With the improvements made in more recently launched phones, we only expect these numbers to be significantly better on them.

Chapter 5

Conclusions

In this thesis we propose a smartphone based indoor localization system which combines the absolute positioning from Bluetooth RSSI based localization with the smooth relative positioning from IMU based inertial odometry and map information to fix drift. We deployed the deep models to mobile phones in the form of iOS and Android apps to run real-time localization thereby facilitating path planning and turn-by-turn navigation.

5.1 Limitations

The biggest limmitation of our approach is generalizability, i.e. being able to use a single trained model in different buildings. Due to the nature of magnetometer data, RSSI data and the use of particle filter to combine them, our models need to be trained on a per building basis. While non-trivial, making these localization models generalize to different buildings should be possible and an interesting avenue to explore.

A key requirement of our system is an accurate map of the indoor space with correct transformations between real-world and map coordinates. Architectural plans, like those used in this work, are not necessarily accurate to the built configuration of a space. Indoor mapping systems based on SLAM using cameras or LiDARs have

become much easier to use and more accurate, but there is still significant effort required to generate maps if they are not previously available for a given space. Both 2D architectural and SLAM maps do not necessarily capture all obstacles in a map, depending on the height of the sensor used to collect the map. For example, tables are not usually placed in their true locations in architectural plans, and a 2D LiDAR system held by a human operator performing mapping of a space may be held too high or low to accurately capture the tables. Obstacle configurations can also change over time without warning. Since our method relies so heavily on combining obstacle information with user motion to rule out areas where the motion was infeasible, the lack of a full understanding of obstacle locations can limit the effectiveness of our approach.

5.2 Future Work

As mentioned in 5.1, trying to make the localization models generalize to multiple buildings is the most important direction that needs to be explored. Another direction to explore is data simulation. Current methods, including ours, require a significant amount of data per building to be able to learn the motion models. Related works mostly in image based localization have shown that it is possible to generate synthetic data from a small amount of actual data and then train localization models to have high accuracy. However, not much study has been done to model RSSI signals or IMU signals for augmentation purposes. Lastly, using differentiable modules instead of the traditional Kalman filter and particle filters we have used in this work would facilitate training an end-to-end sensor fusion model. Use of such approaches in other tasks have shown that they have the potential to be faster and more generalizable while also being more accurate.

Bibliography

[1] J. Wang and D. Katabi. "Dude, Where's My Card? RFID Positioning That Works with Multipath and Non-Line of Sight". In: *Computer communication review* 43.4 (2013), pp. 51–62.

[2] S. Gezici et al. "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks". In: *IEEE Signal Processing Magazine* 22.4 (July 2005), pp. 70–84. ISSN: 1558-0792. DOI: 10.1109/MSP.2005.1458289.

[3] B. Ferris, D. Haehnel, and D. Fox. "Gaussian Processes for Signal Strength-Based Location Estimation". In: *Proceedings of Robotics: Science and Systems*. Philadelphia, USA, Aug. 2006. DOI: 10.15607/RSS.2006.II.039.

[4] S. Hilsenbeck et al. "Graph-Based Data Fusion of Pedometer and WiFi Measurements for Mobile Indoor Positioning". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '14. Seattle, Washington: Association for Computing Machinery, 2014, pp. 147–158. ISBN: 9781450329682. DOI: 10.1145/2632048.2636079. URL: https://doi.org/10.1145/2632048.2636079.

[5] F. Subhan et al. "Indoor positioning in Bluetooth networks using fingerprinting and lateration approach". In: *2011 International Conference on Information Science and Applications* (2011), pp. 1–9.

[6] R. Faragher and R. Harle. "Location fingerprinting with bluetooth low energy beacons". In: *IEEE journal on Selected Areas in Communications* 33.11 (2015), pp. 2418–2428.

[7] E. Mok and G. Retscher. "Location determination using WiFi fingerprinting versus WiFi trilateration". In: *Journal of Location Based Services* 1.2 (2007), pp. 145–159.

[8] H. Liu et al. "Push the limit of WiFi based localization for smartphones". In: *Proceedings of the 18th annual international conference on Mobile computing and networking*. 2012, pp. 305–316.

[9] J. Biswas and M. Veloso. "Wifi localization and navigation for autonomous indoor mobile robots". In: *2010 IEEE international conference on robotics and automation*. IEEE. 2010, pp. 4379–4384.

[10] J. Letchner, D. Fox, and A. LaMarca. "Large-Scale Localization from Wireless Signal Strength". In: *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1*. AAAI'05. Pittsburgh, Pennsylvania: AAAI Press, 2005, pp. 15–20. ISBN: 157735236x.

[11] J. Krumm and E. Horvitz. "LOCADIO: inferring motion and location from Wi-Fi signal strengths". In: *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.* Aug. 2004, pp. 4–13. DOI: 10.1109/MOBIQ.2004.1331705.

[12] P. Bahl and V. N. Padmanabhan. "RADAR: an in-building RF-based user location and tracking system". In: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*. Vol. 2. Mar. 2000, 775–784 vol.2. DOI: 10.1109/INFCOM.2000.832252.

[13] V. Vivekanandan and V. W. Wong. "Concentric anchor-beacons (CAB) localization for wireless sensor networks". In: *2006 IEEE International Conference on Communications*. Vol. 9. IEEE. 2006, pp. 3972–3977.

[14] C. Liu et al. "Range-free sensor localisation with ring overlapping based on comparison of received signal strength indicator". In: *IJSNet* 2 (Jan. 2007), pp. 399–413. DOI: 10.1504/IJSNET.2007.014364.

[15] H. Liu et al. "Survey of Wireless Indoor Positioning Techniques and Systems". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications*

and Reviews) 37.6 (Nov. 2007), pp. 1067–1080. ISSN: 1558-2442. DOI: 10.1109/TSMCC.2007.905750.

[16] M. Altini et al. "Bluetooth indoor localization with multiple neural networks". In: *IEEE 5th International Symposium on Wireless Pervasive Computing 2010*. May 2010, pp. 295–300. DOI: 10.1109/ISWPC.2010.5483748.

[17] W. Zhang et al. "Deep neural networks for wireless localization in indoor and outdoor environments". In: *Neurocomputing* 194 (2016), pp. 279–287.

[18] D. L. Hall, R. M. Narayanan, and D. M. Jenkins. "SDR Based Indoor Beacon Localization Using 3D Probabilistic Multipath Exploitation and Deep Learning". In: *Electronics* 8.11 (2019), p. 1323.

[19] C. Xiao et al. "3-D BLE Indoor Localization Based on Denoising Autoencoder". In: *IEEE Access* 5 (2017), pp. 12751–12760. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2720164.

[20] S. Bai et al. "DL-RNN: An Accurate Indoor Localization Method via Double RNNs". In: *IEEE Sensors Journal* 20.1 (Jan. 2020), pp. 286–295. ISSN: 2379-9153. DOI: 10.1109/JSEN.2019.2936412.

[21] K. Urano et al. "An End-to-End BLE Indoor Location Estimation Method Using LSTM". In: *2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*. Nov. 2019, pp. 1–7. DOI: 10.23919/ICMU48249.2019.9006638.

[22] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. "Estimation of IMU and MARG orientation using a gradient descent algorithm". In: *2011 IEEE International Conference on Rehabilitation Robotics*. 2011, pp. 1–7. DOI: 10.1109/ICORR.2011.5975346.

[23] W. S. Beauregard and M. Klepal. "Indoor PDR performance enhancement using minimal map information and particle filters". In: *2008 IEEE/ION Position, Location and Navigation Symposium*. 2008, pp. 141–147. DOI: 10.1109/PLANS.2008.4570050.

[24] A. M. C. Chen. X. Lu and N. Trigoni. "IONet: Learning to Cure the Curse of Drift in Inertial Odometry". In: *CoRR* abs/1802.02209 (2018).

[25] H. Yan, S. Herath, and Y. Furukawa. "RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, and New Methods". In: *CoRR* abs/1905.12853 (2019).

[26] W. Liu et al. "TLIO: Tight Learned Inertial Odometry". In: *CoRR* abs/2007.01867 (2020).

[27] S. Sun, D. Melamed, and K. Kitani. "IDOL: Inertial Deep Orientation-Estimation and Localization". In: *CoRR* abs/2102.04024 (2021).

[28] S. Herath et al. "Neural Inertial Localization". In: *arXiv preprint arXiv:2203.15851* (2022).

[29] A. Luo, S. Chen, and B. Xv. "Enhanced Map-Matching Algorithm with a Hidden Markov Model for Mobile Phone Positioning". In: *ISPRS International Journal of Geo-Information* 6.11 (2017). ISSN: 2220-9964. DOI: 10.3390/ijgi6110327. URL: https://www.mdpi.com/2220-9964/6/11/327.

[30] K. Zhao et al. "DeepMM: Deep Learning Based Map Matching with Data Augmentation". In: *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '19. Chicago, IL, USA: Association for Computing Machinery, 2019, pp. 452–455. ISBN: 9781450369091. DOI: 10.1145/3347146.3359090. URL: https://doi.org/10.1145/3347146.3359090.

[31] Z. Xiao et al. "Lightweight map matching for indoor localisation using conditional random fields". In: *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*. 2014, pp. 131–142. DOI: 10.1109/IPSN.2014.6846747.

[32] J. S. Liu and R. Chen. "Sequential Monte Carlo Methods for Dynamic Systems". In: *Journal of the American Statistical Association* 93.443 (1998), pp. 1032–1044. ISSN: 01621459. URL: http://www.jstor.org/stable/2669847 (visited on 04/11/2022).

[33] R. A. MacLachlan and A. Dubrawski. "Applied Indoor Localization: Map-based, Infrastructure-free, with Divergence Mitigation and Smoothing". In: *Proceedings*

of 13th International Conference on Information Fusion (FUSION '10). Vol. 794. July 2010, pp. 801–802.

[34] H. Xia et al. "Indoor Localization on Smartphones Using Built-In Sensors and Map Constraints". In: *IEEE Transactions on Instrumentation and Measurement* 68.4 (2019), pp. 1189–1198. DOI: 10.1109/TIM.2018.2863478.

[35] A. Rechy Rormero et al. "Map-Aware Particle Filter for Localization". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 2940–2947. DOI: 10.1109/ICRA.2018.8460707.

[36] C. Peng and D. Weikersdorfer. "Map as The Hidden Sensor: Fast Odometry-Based Global Localization". In: *CoRR* abs/1910.00572 (2019).

[37] S. Herath et al. "Fusion-DHL: WiFi, IMU, and Floorplan Fusion for Dense History of Locations in Indoor Environments". In: *CoRR* abs/2105.08837 (2021).

[38] M. Mohammadi et al. "Semi-supervised Deep Reinforcement Learning in Support of IoT and Smart City Services". In: *CoRR* abs/1810.04118 (2018). arXiv: 1810.04118. URL: http://arxiv.org/abs/1810.04118.

[39] X. Zhao et al. "Does BTLE measure up against WiFi? A comparison of indoor location performance". In: *European Wireless 2014; 20th European Wireless Conference*. May 2014, pp. 1–6.

[40] J. Zhang and S. Singh. "LOAM: Lidar Odometry and Mapping in Real-time." In: *Robotics: Science and Systems*. Vol. 2. Berkeley, CA. 2014, pp. 1–9.

[41] V. V. Lehtola et al. "Comparison of the selected state-of-the-art 3D indoor scanning and point cloud generation methods". In: *Remote sensing* 9.8 (2017), p. 796.

[42] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv:1505.04597 [cs]* (May 2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (visited on 06/03/2021).

[43] S. Padhy. *shreyaspadhy/UNet-Zoo*. original-date: 2017-12-09T20:48:15Z. May 2021. URL: https://github.com/shreyaspadhy/UNet-Zoo (visited on 06/03/2021).

[44] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. OCLC: ocm58451645. Cambridge, Mass: MIT Press, 2005. ISBN: 978-0-262-20162-9.