A PROBABILISTIC FRAMEWORK FOR MULTI-SENSOR FUSION BASED INDOOR LOCALIZATION ON MOBILE PLATFORM

by

XIANG HE

A dissertation submitted in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL AND COMPUTER ENGINEERING

2017

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Jia Li, Ph.D, Chair
Daniel N. Aloi, Ph.D
Daniel Steffy, Ph.D
Tao Shu, Ph.D

This dissertation is dedicated to my father Songqiang He, my mother Xiaona Liu and my wife Chenwen You.

ACKNOWLEDGEMENTS

It has been six years since I first landed on DTW airport. I came to the United States to chase my dream of getting a Ph.D. degree. It has been an incredible journey and I would like to share my gratefulness with those who help me to get to this end.

I would like to first thank my advisor, Dr. Jia Li, for her continuous encouragement, expert advice and support of my dissertation research. Her profound knowledge and expertise on digital signal processing, machine vision and statistics modeling are sources I can always rely on.

I would like to acknowledge my Ph.D. committee members, Dr. Daniel N. Aloi, Dr. Daniel Steffy and Dr. Shu Tao for their advices and time during my Ph.D. study and research.

I would also like to express my gratitude to my former colleagues Juan Li and Kapil Mistry for their help on LiDAR and camera sensor data acquisition system setup and providing me with the source codes of the image processing algorithms in MATLAB.

Finally, I am grateful to my parents for their constant support and encouragement. Without their support, this dissertation work is impossible. I would also like to thank my wife, Chenwen You. Her love and patience have always been my motivation to commit excellence in my research.

This research project is supported by Fiat Chrysler Automobiles research fund on project titled “Location Awareness of Portable Pendant”.

Xiang He

ABSTRACT

A PROBABILISTIC FRAMEWORK FOR MULTI-SENSOR FUSION BASED INDOOR POSITIONING SYSTEM ON MOBILE PLATFORM

by

Xiang He

Adviser: Jia Li, Ph.D.

Nowadays, smart mobile devices integrate more and more sensors on board, such as motion sensors (accelerometer, gyroscope), wireless signal strength indicators (WiFi, Bluetooth), and visual sensors (LiDAR, camera). People have developed various indoor localization techniques based on these sensors. In this dissertation, a probabilistic framework for multi-sensor fusion based indoor localization system is developed and partially implemented on a mobile platform.

The probabilistic fusion of multiple sensors is investigated in a hidden Markov model (HMM) framework for mobile device user localization. We propose a graph structure to store the model constructed by multiple sensors during offline training phase, and a multimodal particle filter to seamlessly fuse the information during online tracking phase.

The multi-sensor information for our data fusion and analysis includes WiFi received signal strength (RSS) collected from mobile device's received signal strength indicator (RSSI), motion signals gathered by built in motion sensors including accelerometer and gyroscope, and images captured by camera.

Based on our algorithms, we performed simulations in MATLAB and analyzed the results. We further implemented the indoor localization system on the iOS platform. The experiments carried out in typical indoor environment have shown promising results of the proposed algorithm and system design.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Problem Statement

Nowadays, smart mobile devices (smartphone, tablet or wearable device like glasses or watch) are playing more and more important role in people's daily life. In some sense, they have become a new organ of human being. The ever-improving innovative technology is extending the function of our eyes, ears and mouths. It is changing human DNA in a digital way. In this dissertation, we focus on one functionality extension for human sense: localization. It is well known that a bat is able to determine the distance of objects from the time between expelling and hearing the ultra-sound emitted by itself. This is called echolocation. Human doesn't gain this ability during our evolution, but we are smart enough to build machines to improve our sense of location. The most famous one is the Global Positioning System (GPS). By receiving location information from satellites, a handheld receiver is able to localize the user on earth with meter level accuracy. However, GPS may not function in an indoor environment due to weak or complete loss of signals.

In recent years, researchers have developed various approaches for mobile device user localization in GPS-denied indoor environment. To name a few, radio frequency (RF) fingerprinting techniques, motion sensor based pedestrian dead reckoning (PDR) techniques and visual sensor based feature matching techniques are the most popular approaches in indoor localization. However, all of them have their own limitation. RF fingerprinting techniques based on Bluetooth [1], RFID [2], Zigbee [3] or WiFi [4] [5]

[6],) have the problem of signal fluctuation due to the multipath fading effect in an indoor environment. The motion sensor based PDR approach [7] [8] suffers from the fact that the motion sensors equipped on the mobile device are low cost Micro Electromechanical System (MEMS) sensors, which have relatively low accuracy. Thus the integration drift will cause the localization deviation to accumulate over time. The visual sensor based localization techniques [9] [10] [11] [12] [13] extract visual features like SIFT [14] or SURF [15] from captured images and compare them with an image database. The costly feature matching algorithm and restricted computation resource on mobile platform limits the deployment of such techniques. Moreover, in ASSIST [16], inspired by bat echolocation mechanism, acoustic signals emitted from smartphone speakers are adopted to locate the user using time differences of arrival (TDoA) multilateration method. ASSIST can locate the user within 30 cm. However, this method requires sound receivers preinstalled at the ceiling or walls, which adds extra infrastructures to the indoor environment.

To overcome the drawback of each sensor, people have come up with fusion approach to combine different sensors to achieve better localization result. However, since the sensors are measuring different physical phenomena, it's not an easy task to effectively fuse the information from multiple sensors. The existing sensor fusion approaches for localization involve decision level fusion and feature level fusion. The decision level fusion usually contains multiple local detectors and a fusion center. The local decisions are transmitted to the fusion center where the global decision is derived. The optimum decision rule under Neyman-Pearson criteria can be expressed as a function of correlation coefficients of local detectors. It has been shown that the performance of

such distributed detection system degrades as the degree of correlation increases [17]. This approach is easy to implement and computationally efficient. It has been widely used in wireless sensor network (WSN) and some other research fields [18]. However, it is not practical in the indoor localization system with multiple sensors due to the difficulty of determining the correlation coefficients between different sensors. On the other hand, feature level fusion [19] is a more delicate fusion approach that extracts features from multiple sensor observation, and uses these features to represent the real world and help localization. The problem of feature level sensor fusion is the highly redundant sensor data in feature extraction. As there are multiple sensors, each sensor delivers different data about the surrounding environment, we have to determine an effective approach to extract the information and store them in an efficient way so that we can access them easily for purpose of localization. Existing fusion algorithms include Bayesian filtering techniques, such as Kalman filter [20] [21] and particle filter [22] [23], and non-Bayesian filtering technique, for example, conditional random field [24] [25] and Dempster-Shafer theory [26] [27]. Originally, Kalman filter and particle filter are designed for state estimation in single-sensor measurements. But information fusion based on Bayesian filtering theory has been studied and widely applied to multi-sensor systems. Generally speaking, there are two types of methods to process the measured sensor data. The first one is the centralized filter where all sensor data are transferred to a fusion center for processing. The second one is the decentralized filter where the information from local estimators can achieve the global optimal or suboptimal state estimate according to certain information fusion criterion.

1.2 Our Contributions

In this dissertation, we first adapt Gaussian process modeling of WiFi RF fingerprinting and particle filter based localizer to mobile device [28]. Then we introduce the motion sensors on board to improve the localization accuracy [29]. The algorithm has been implemented on iOS platform and tested in an indoor environment. To further improve the localization accuracy, we introduce visual sensors into the system [30]. The probabilistic model for multi-sensor fusion is investigated in a hidden Markov Model (HMM) framework, where the state transition model is defined as the user motion model, and the observation model includes WiFi sensor model, camera sensor model and motion sensor model [31]. Researchers have applied HMM successfully in WSN area. Huang et al modeled the dynamic quantization and rate allocation in a sensor network with a fusion center as a finite state Markov chain, and designed an optimal quantizer using stochastic control approach for state estimation in hidden Markov model [32]. Rossi et al developed a HMM framework which exploits time-correlation of the unknown binary source under observation through a WSN reporting local sensor detection to a fusion center over Rayleigh fading channel [33]. To solve the HMM state estimation problem with multiple sensors, Blom et al proposed the interacting multiple model (IMM) algorithm, which combines state hypotheses from multiple filter models to get a better state estimate of targets with changing dynamics [34]. The filter models used to form each state hypothesis can be derived to match the targets of interest's behavior. In this dissertation, we propose a multimodal particle filter to seamlessly fuse the data from multiple sensors for HMM state estimation.

A graph structure $G = (V, E)$ is developed to store both the user motion information and environmental information effectively. The key idea is to represent the indoor environment by graph whose vertices correspond to segments of the indoor environment. The segments are predefined in a 3D model built offline. The vertices play an important role in the motion model since they relate to movement choices, which are a limited amount of positions user can move to next. The edges correspond to connection between different segments, which act as constraints of the user movement to reduce the computation during the online tracking phase.

Specifically, we made the following contributions to the field of indoor localization. Under the HMM framework, we propose a graph structure to store the model constructed by multiple sensors in offline training phase, and a multimodal particle filter to fuse the information efficiently during online tracking phase. The particle filter is able to handle the motion sensor drift problem during the resampling step. The WiFi signal strength fluctuation problem is mitigated using the motion sensor information to guide the particle propagation towards the higher likelihood field. We developed an indoor localization system on iOS platform. To the best of our knowledge, our iOS application is the first one to achieve accurate, robust, and highly integrated indoor localization by seamlessly fusing the information from multiple sensors on board.

1.3 List of Publications

1.3.1 Journal

1. X. He, D. N. Aloi, J. Li, "Probabilistic multi-sensor fusion based indoor positioning system on a mobile device" in *Sensors*, December 2015, vol. 15, pp. 31464-31481.

1.3.2 Peer-reviewed Conference Proceedings

1. X. He, S. Badiei, D. N. Aloi, J. Li, "WiFi iLocate: WiFi based indoor localization for smartphone" in *Proceedings of Wireless Telecommunication Symposium*, Washington, DC, USA, August 2014, pp. 1-7.

2. X. He, D. N. Aloi, J. Li, "WiFi based indoor localization with adaptive motion model using smartphone motion sensors" in *Proceedings of the International Conference on Connected Vehicle and Expo (ICCVE)*, Vienna, Austria, November 2014, pp. 786-791.

3. J. Li, X. He, J. Li, "2D LIDAR and camera fusion in 3D modeling of indoor environment" in *Proceedings of National Aerospace and Electronics Conference (NAECON)*, Dayton, USA, June 2015, pp. 379-383.

4. X. He, D. N. Aloi, J. Li, "Portable 3D visual sensor based indoor localization on mobile device" in *Proceedings of the Consumer Communication & Networking Conference (CCNC)*, Las Vegas, USA, January 2016, pp. 1125-1128.

1.4 Dissertation Outline

The remainder of this dissertation is organized as follows. In the next chapter, various indoor localization systems and related localization algorithms are introduced. In Chapter Three, we give an overview of multi-sensor fusion techniques. In Chapter Four, a WiFi based localization system applying Gaussian process modeling of the WiFi RSS, including the simulation results in MATLAB and real time testing on iOS device are described. We improve the localization system performance by fusing motion sensor information from the mobile device. The result and analysis are discussed in Chapter Five. Then we discuss the application of using visual sensors for indoor localization on mobile device in Chapter Six. In Chapter Seven, we introduce a probabilistic framework, which

is able to fuse the information from various sensors for indoor localization. The details of offline training, HMM modeling and online tracking of our algorithms are described. Finally, Chapter Eight summarizes the research with a discussion of future work.

CHAPTER TWO

TECHNIQUES FOR INDOOR LOCALIZATION SYSTEMS

Localization systems, no matter what type of information they are based upon, need to obtain data from the surrounding environment with sensors input. Thus, sensing is the first step to begin a localization task, which is followed by processing and analyzing the obtained data with different kinds of algorithms. In this chapter, we will first introduce some wireless signal based localization methods. Then we will take a look at the localization system based on motion sensors. After that, we will turn to visual sensors to see how they can be applied to localization and tracking tasks.

2.1 Wireless Signal Based Localization

Researchers have experimented with various systems using wireless signal to locate and track user position. The most widely applied localization systems are based on Bluetooth, RFID and WiFi signals.

2.1.1 Bluetooth and RFID Based Localization

The Bluetooth technology was originally proposed for short distance data transfer. It is also capable of reporting the distance between the transmitter and the receiver. The Bluetooth based indoor localization system usually involves two parts: the Bluetooth beacons, which are low cost, low energy chips and can last for years, and the mobile receivers, which are usually smartphones, like iPhones or Android phones. The smartphone receives a signal from these beacons, and calculates the current location using triangulation algorithm. There are several companies providing the Bluetooth

beacon based indoor localization services. For example, SPREO develops a Bluetooth beacon signal based indoor navigation system, and claims an accuracy of 1.5m [35].

The Radio Frequency Identification (RFID) based indoor localization method is another type. The idea is to use the electromagnetic field to wirelessly transfer data to automatically identify and track tags attached to objects. There is a bidirectional communication between a sender and a tag. Tags can be active, battery-assisted passive or passive. The active tags have built-in batteries and can periodically transmit signals. The battery assisted passive tags have a small battery inside but the battery is only activated when the tags are in the range of a reader. The passive tags are small and cheap, but they require a signal magnitude three times than the normal because they need to use the signal energy to transmit the data back to the reader. Comparing to barcodes, RFID tags do not necessarily have to be within line of sight of a RFID reader, and may be embedded in a tracked object. So it is consider as a potential solution in retail stores for product identification. In Chumkamon's work [36], the RFID tags are embedded into stone blocks and put onto a footpath. By continuously detecting and tracking the RFID signals with a portable RFID reader, the app can determine the current location by decoding the received signals and looking it up in the database.

Both the Bluetooth and RFID localization methods require modification to the environment by attaching beacons or tags in order to function properly. When the area of localization service expands, the cost would increase significantly, and therefore limit the scalability of these methods. Meanwhile, changing the environment setting would cause other issues, such as aesthetic issues or legal issues. In addition, the accuracy of the

localization depends heavily on the number of sensors (beacons or tags) mounted, which adds to the burden of installation and maintenance of these systems.

2.1.2 WiFi Based Localization

Generally speaking, WiFi based indoor localization techniques can be categorized in two types, propagation based and location fingerprinting based. Propagation based algorithms usually apply mathematical models to a set of triangulation algorithms to determine the location of the device. The triangulation approach uses the geometric properties of triangles to estimate the target location. Specific techniques use information include angle of arrival (AOA), phase of arrival (POA), time of arrival (TOA) and roundtrip time of flight (RTOF) to localize the device [37]. The main drawback of the propagation based algorithms is the difficulty in getting an accurate propagation model for the complicated indoor environment. Due to this difficulty, propagation based techniques can only achieve limited accuracy.

Instead of modeling the propagation of WiFi signal, location fingerprinting based algorithms assumes that a WiFi enabled device always receives similar signal strength at a certain location, such received signal strength (RSS) and coordinates would serve as a unique “fingerprint” of this location. We can collect a “fingerprint” in each location and store them in a dataset. Every time user comes to a new location, the WiFi signal strength is detected and the location is estimated by measuring the similarity between current and stored fingerprints.

There are many location fingerprinting based methods, such as the K-Nearest Neighbor (K-NN), neural networks, support vector machine (SVM) and the probabilistic method [37]. The K-NN algorithm compares the online scanned WiFi RSS with the

offline built WiFi RSS dataset, searches for K closest matches RSS values in signal space, and uses these K known locations to estimate the current location. K-NN is easy to implement but it suffers greatly from signal fluctuations as the RSS detected at the same location may vary from time to time. Neural Network and SVM both are machine-learning methods used for classification and regressions. They require huge amount of training data and complex training process to achieve high accuracy result, which are not applicable in mobile device. By introducing Gaussian process into our system, we are able to build an accurate WiFi RSS model without the need of intensive survey data collection and sophisticated training process. The Gaussian process modeling will be discussed in detail in Chapter Four.

2.2 Motion Sensor Based Localization

Motion sensors have been around for decades. Traditionally they have been used in aviation and marine industry, where they are used for navigation and control purposes. They are able to determine the instantaneous location and orientation of a platform both accurately and rapidly. However, these kinds of motion sensors are usually bulky and pretty expensive.

The development of micro machined electromechanical system (MEMS) technology has leaded to smaller and cheaper motion sensors. Nowadays motion sensors have become more and more ubiquitous. They can be found in cars, gaming consoles and in smart mobile device like iPhone or tablet. The functionality of MEMS motion sensors is based upon simple mechanical principles. For example, angular velocity can be measured by exploiting the Coriolis ffect of a vibrating structure. When a vibrating structure is rotated, a secondary vibration is induced from which the angular velocity can

be calculated. Acceleration can be measured with a spring suspended mass. When subjected to acceleration the mass will be displaced.

Base on motion sensor measurements, researchers have developed pedestrian dead reckoning (PDR) approaches for indoor localization. It is a process of calculating one's current location by using a previous determined location, and advancing that location based on estimated speeds over a certain time period. It thus requires the user to manually input the initial starting point. According to the place of the installed sensors, PDR can be categorized as foot-mounted [38] [39] [[40] [41], waist-mounted [42] [43], and handheld [44] [45] [46] types. Foot-mounted and waist-mounted PDR usually require a customized sensor to wear on. It can provide accurate step length estimation result, but cannot solve the heading problem. Handheld type PDR usually refers to a smartphone application. In Li et al's work [7], they designed reliable algorithms for pedestrian step detection, stride estimation and heading inference using motion sensors (accelerometer, gyroscope and magnetometer) built in the smartphone, and achieve meter-level accuracy in indoor environment.

The biggest challenge for motion sensor based localization is the internal defect of the low cost MEMS sensors. The integration drift will cause the localization deviation to accumulate over time. Thus, PDR is better to be used as a supplement to other method like WiFi based localization system. We will explain our usage of motion sensor data in Chapter Five.

2.3 Visual Sensor Based Localization

Naturally, human beings tend to rely on visual information to perceive the environment and perform daily tasks. Visual sensor based approaches are drawing more

and more attention in both the computer vision community and the robotics community to achieve real time and accurate localization system. Depending on how to use the data captured by visual sensors, we can classify vision based localization methods to two categories: 2D approaches and 3D approaches.

2.3.1 2D Image Based Localization

Two-dimensional (2D) image-based approaches directly use original 2D images, transformations of input images, or their 2D features, to localize the images. Typically, a 2D method avoids 3D inference and motion parallax problems for navigating a 3D scene by using only 2D image features; and the model of the 3D scene is usually a database of images or features, organized with geo-location tags, instead of building and maintaining a 3D model.

The main idea of 2D image-based localization is to first store the scene images, or features of these images, into a database, indexed and related with known geographical locations. A new input image to be localized is preprocessed with the same kind of feature extraction procedure and is retrieved in the database. Since the images in the database are geo-located, finding the matched images are equivalent to successfully localizing the input image.

The authors in [47] take advantage of off-the-shelf 2D image features, namely color histogram, wavelet decomposition and shape information to achieve room level accuracy with over 90% success probability. However, this approach cannot be used to determine the absolute position of the camera, nor its orientation. Thus limits its application where precise position and orientation are needed.

Researchers demonstrate an image based localization system for mobile devices, which achieves not only sub-meter localization accuracy but also determines orientation in [48]. They use a human operated ambulatory backpack to collect geo-tagged images with SIFT features and associated 6 degrees of pose of all images. Then they load all of the image database SIFT features into a KD-Tree and perform K-Nearest Neighbor (KNN) search to find a database image with most number of matching features to the query image. Then they apply the SIFT features match along with mobile device pitch and roll to recover the relative pose between the retrieved database image and the query image. This results in 6 degree of freedom pose for the query image.

2.3.2 3D Model Based Localization

We live in a 3D space. So it is more natural to sense the world with 3D sensors. There are quite a few methods to obtain the third dimension data - the depth. Here we only discuss two major method types: Structure from Motion (SfM) methods and direct 3D sensing methods.

SfM is the process of estimating 3D structure from 2D image sequences, which can be describe as the following steps: 1. Record a video that captures every details of the environment. The video is served as input to the SfM process. 2. Perform SIFT feature extraction and pairwise feature matching among the sequential images from input video. 3. Reconstruct the 3D model by estimating the 3D position of visually relevant points in the scene, as well as the position and orientation of the cameras from where the pictures are taken. 4. Refine the 3D structure using nonlinear optimization methods to minimize the reprojection error of all the reconstructed 3D points to the 2D image position of the original SIFT features in the images. Ruiz et al use the SfM based 3D model combining

with WiFi signals and motion sensor data to build a Location based Service (LBS) system that achieve real time (0.5 seconds of response time) and high accuracy (average error around 15 cm) [49].

The 3D environment information can also be obtain directly with portable RGB-Depth sensors, such as Microsoft's Kinect, Google Tango Tablet or Occipital's Structure Sensor for iPad. These devices are able to register the color images with the depth images. Then automatically or manually aligning individual, consecutive local 3D point clouds can generate a complete 3D model. As describe in [50], we can perform localization by comparing the online captured 3D point cloud with the offline build 3D model to determine the actual pose of the mobile device. Since we have the 3D coordinates of the key frame image's feature points, given in its registered depth image, and the 2D coordinates of the captured image's corresponding feature points, specified in the image, these N feature points form a classic Perspective-n-Point (PnP) problem. By solving the PnP problem, we get the transformation matrix between the captured image and the key frame image in the database. Therefore, the initial 6 degree-of-freedom pose can be calculated by multiplying the global pose of the key frame image with the transformation matrix. Once the initial pose is known, the ongoing pose estimation problem has become finding the rigid transform between the online captured 3D point clouds with the offline built 3D model, which can be solved with iterative closest point (ICP) algorithm. We will further discuss this method in Chapter Six.

CHAPTER THREE

MULTI-SENSOR FUSION TECHNIQUES OVERVIEW

Multi-sensor fusion is a process of association, correlation and combination of information collected from various sources into one framework that can help a system to make better decision than from single source only. In order to accomplish this task, a mathematical framework must be used to combine these multiple sensor readings to a single decision output. In this chapter, we will have a review of different kinds of sensor fusion techniques.

3.1 Probabilistic Fusion

Generally speaking, sensor fusion techniques perform a data mapping from multiple inputs to a smaller number of outputs. Sensor inputs usually include raw sensor measurements, like pixel values, received signal strengths, or accelerated velocities. Outputs can be an estimated state, a recognized objects or a decision. Probabilistic fusion methods use probability density function to express sensor data uncertainty. By combining prior and observation information, we are able to reach a posterior probability for the decision or estimation problem. At the core of these methods lies the Bayes rule, which provides a way to infer about an object of interest described by a state x, given an observation z.

3.1.1 Bayesian Inference

British scientist Thomas Bayes discovered Bayesian Theory in 1763. The well-known Bayes rule describes the fundamental probability law governing the logical inference process. In Bayes theory, the posterior probability depends on a likelihood

function and a prior probability. According to Bayes formula, the posterior probability is calculated as follows:

$$p(x \mid z) = \frac{p(z \mid x)p(x)}{p(z)} \tag{1}$$

In the above equation, the prior beliefs about what values of x might be expected, is encoded in the prior probability $p(x)$. To obtain more information about the state x, an observation z is made. The observations are modeled as a conditional probability $p(z \mid x)$, which describes the probability of making an observation of z given a state x. The new likelihood associated with the state x is calculated from the product of the original prior information and the information gained by observation. This is encoded in the posterior probability $p(x \mid z)$. The marginal probability $p(z)$ serves as a normalization term for the posterior.

The multisensory form of Bayes rule requires conditional independence:

$$p(x \mid Z^n) = Cp(x)\prod_{i=1}^{n} p(z_i \mid x) \tag{2}$$

where C is a normalizing constant. z_i means the observation from sensor i. Z^n represents all observations from n sensors.

This states that the posterior probability of x given observations Z^n is proportional to the product of prior probability and individual likelihoods from each information source.

In Bayesian inference, the uncertainties in sensor information are modeled as random variables. The inference is performed within the Bayesian framework given all of available information. The goal of Bayesian inference is to use prior knowledge and finite

observations from sensors to infer the conditional probability. There are usually three levels of probabilistic reasoning in Bayesian analysis: (1) Model selection given the data and assumed priors; (2) Estimating the parameters to fit the data given the model and priors; (3) Updating the parameters. Bayesian inference can be illustrated by a directed graph, a Bayesian network, which is a probabilistic graphical model with a set of vertices and edges, the probability dependency is described by a directed arrow between two nodes that represent two random variables. Hidden Markov model (HMM) is one type of Bayesian network and we will have a further discussion of it in Chapter Four.

3.1.2 Bayesian Filtering

In order to do state estimation over time, researchers have come up with the recursive Bayesian estimation, also known as Bayesian filtering. The recursive form of Bayes rule is:

$$p(x \mid Z^k) = \frac{p(z_k \mid x)p(x \mid Z^{k-1})}{p(z_k \mid Z^{k-1})} \tag{3}$$

where z_k is an observation at time k. Z^k represents the observations up to time k.

The advantage of recursive Bayesian formula is that we only need to compute and store the posterior probability $p(x \mid Z^{k-1})$, which contains a complete summary of all past information. When the next piece of information $p(z_k \mid x)$ arrives, the previous posterior takes on the role of the current prior and the product of the two becomes the new posterior after normalization.

The most popular Bayesian filtering technique is the Kalman filter. It has a number of features that make it suitable to deal with complex multi-sensor estimation and data fusion problems. The explicit description of process and observations allows a wide

variety of different sensor models to be incorporated within the basic algorithm. In addition, the consistent use of statistical measures of uncertainty makes it possible to quantitatively evaluate the role each sensors play in overall system performance. Further, the linear recursive nature of the algorithm ensures that its application is simple and efficient. For these reasons, the Kalman filter has found wide spread application in many different sensor fusion problems [51] [52]. On the other hand, Kalman filter is limited to linear models with additive Gaussian noises.

When dealing with non-linear system dynamics, one usually has to resort to approximation techniques. Extended Kalman filter (EKF) [53] and Unscented Kalman filter (UKF) [54] are based on first order and second order approximation in Taylor series expansion. However, both of the methods can only handle non-linearity to a limited extend. Grid-based methods [55] provide an alternative approach for approximating non-linear probability density functions, although they rapidly become computationally intractable in high dimensions. Furthermore, particle filter based on Monte Carlo method has been developed in order to represent the posterior probability in terms of random samples and associated weights [56]. Particle filter does not require the restrictive hypotheses in the Kalman filter. Hence, it can be applied to any non-linear models with non-Gaussian errors. Particle filter is used to approximate the posterior probability of the system state as a weighted sum of random samples. The random samples are drawn from the prior density (state transition model) with their weights updated according to the likelihood of the given measurement (sensor observation model). This approach performs a resampling step where the current set of particles is replaced by a new set drawn from it with probabilities to their weights. This step is called sequential importance resampling

(SIR), which is the key part within particle filtering. We will discuss the particle filter in detail for our system during the online tracking phase.

3.2 Evidence Theory

The theory of evidence was first proposed by Dempster and later on extended by Shafer [57]. The theory models uncertainty as belief in one or more propositions or ignorance. It is based on two ideas: obtaining degrees of belief for one question from subjective probabilities for a related question, and Dempster's rule for combining such degrees of belief when they are based on independent items of evidence. In essence, the degree of belief in a proposition depends primarily upon the number of answers (to the related questions) containing the proposition, and the subjective probability of each answer. Also contributing are the rules of combination that reflect general assumptions about the data.

Mathematically speaking, consider a universal set X. In probability theory, a degree of belief called belief mass may be placed on any element $x_i \in X$ and on any subset $A \in X$. In evidence theory, belief mass not only can be placed on elements and sets, but also sets of sets. Specifically, while X represents all possible states of a system, the power set 2^X represents the set of all possible subsets of X.

As an example, consider the mutually exclusive set $X = \{occupied, empty\}$. In probability theory we might assign a probability to each possible event like this: $p(occupied) = 0.3$, $p(empty) = 0.7$. In evidence theory, we construct the set of all subsets: $2^X = \{\{occupied, empty\}, \{occupied\}, \{empty\}, \phi\}$, and belief mass is assigned to all elements of this set as $m(\{occupied, empty\}) = 0.5$, $m(\{occupied\}) = 0.3$,

$m(\{empty\}) = 0.2$, $m(\phi) = 0.0$. The interpretation of this is that there is a 30% chance of occupied, a 20% chance of empty and a 50% chance of either occupied or empty. In effect, the measure placed on the set containing both occupied and empty, is a measure of ignorance or inability to distinguish between the two alternatives. Thus, evidence theory provides a method of capturing ignorance. In probability theory, this would be dealt with in a very different manner by assigning an equal or uniform probability to each alternative. Yet, stating that there is a 50% chance of occupancy is clearly not the same as saying that it is unknown if it will occupied or not. The use of power set as all possible state of the system allows a far richer representation of beliefs. However, this comes at the cost of a substantial increase in complexity.

Garvey et al [58] first applied evidence theory in data fusion problem in 1981. Unlike the Bayesian inference, the evidence theory allows each source to contribute information in different levels of detail. For example, one sensor can provide information to distinguish individual entities, whereas other sensors can provide information to distinguish classes of entities. The study by Barrnett [59] first addresses the computation problems of implementing Dempster's rule of combination. In his proposed algorithm each piece of evidence either confirms or denies a proposition. He realizes linear time computations and demonstrates the results in experiments.

CHAPTER FOUR

WIFI BASED INDOOR LOCALIZATION AND TRACKING

In this chapter, we will discuss in detail the WiFi based indoor localization techniques. Including the offline training phase, online tracking phase, simulation in MATLAB and implementation on iOS platform.

4.1 Introduction

Nowadays, WiFi access points (APs) have become ubiquitous, whether in the offices, museums, shopping malls or airports. Modern people can hardly survive without access to the Internet. Although it's more like a joke instead of serious theory, people argue that WiFi has made its space in Maslow's hierarchy of basic human needs as shown in Figure 1. In the meantime, smartphones are playing more and more important roles in people's daily life. We often see people with smartphones walking around in the public areas. An accurate indoor localization system can help people easily accessing navigation in a museum or airport terminal, finding specific merchandise or promotion information in a shopping mall, or locating themselves whenever they get lost. Global Positioning System (GPS) is commonly used for navigation outdoors. But it lacks enough accuracy when functioning indoor. People are trying to develop WiFi based Positioning System (WPS) to fulfill the indoor localization task.

A WiFi based localization system has several advantages: First, WiFi APs are becoming a standard configuration in many of the indoor environments. Second, WPS only rely on the existing infrastructure, so no modification to the environment is required. Third, the WiFi information needed for doing localization include only the received.

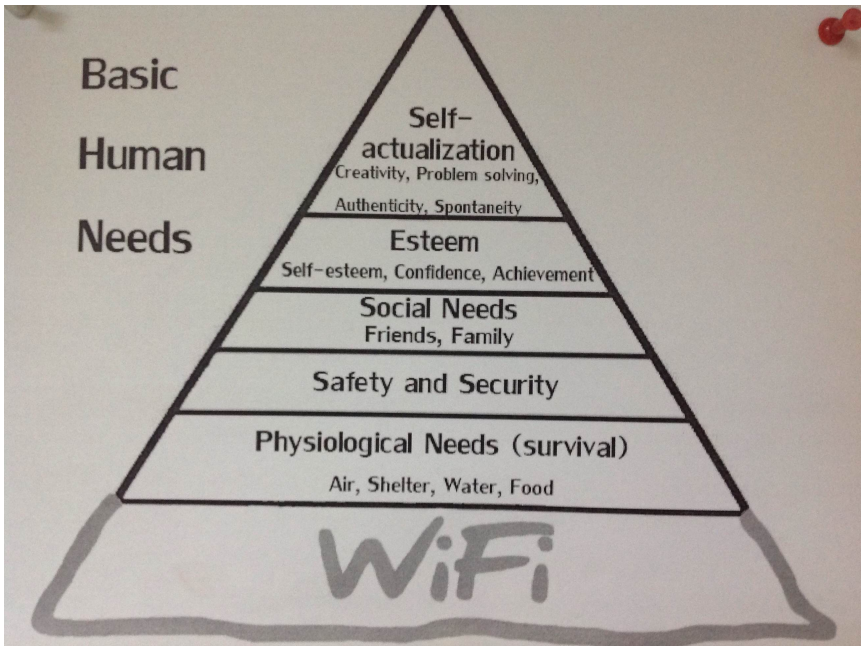


Figure 1: Maslow's hierarchy of basic human needs (image from imgur)

Third, the WiFi information needed for doing localization include only the received signal strength (RSS) and the Basic Service Set Identifier (BSSID). The information is easy to collect, simply by sniffing the wireless traffic in the air. Fourth, WiFi signals do not require line-of-sight (LOS). It is extremely suitable for the use in indoor environment where there are a lot of walls and obstructions.

Extensive research has been done in developing the WiFi based indoor localization systems, such as RADAR [60], Horus [61], Compass [62]. However, most of them are developed on laptop platforms equipped with better antennas than on the smartphone. Moreover, recent work on developing smartphone indoor localization app achieves only room level accuracies, like Shopkick[63]. Therefore, accurate indoor localization on smartphone still remains an open problem. Liu et al [64] tried to solve this

problem by introducing peer assisted localization approach. But this approach only works in public areas with high densities of smartphones present at the same time. Lokesh et al [65] described an accurate smartphone based indoor pedestrian localization system using WiFi and camera on the phone. But they only demonstrated their results in simulation instead of an actual implementation on smartphone.

Several RSS based localization systems that utilize Gaussian process regression have already been developed, and this approach has proved to be well suited in modeling the RSS dataset [66]. But none of them is adapted to the smartphone platform. Our approach is inspired by their work and tailored for the purpose of smartphone application. The two main components of our WiFi based localization and tracking algorithm are a WiFi RSS dataset trained by Gaussian process regression and a localizer based on particle filter.

Specifically, we make the following contributions: We build a WiFi based indoor localization and tracking system on an iOS platform. It does not require any dedicated infrastructure in the indoor area or specialized hardware equipped on the smartphone. Moreover, our system does not require any user-specific information, such as user's initial location or AP's location. It is press-to-go localization. To the best of our knowledge, our iOS application WiFi iLocate is the first one to achieve real time, highly accurate indoor localization by leveraging Gaussian process WiFi RSS fingerprinting modeling along with particle filter based localizer in smartphones.

The remainder of this chapter is organized as follows: In Section 4.2, we give an overview of Gaussian process regression and show how it can be used in modeling WiFi RSS fingerprinting. Then we describe a localizer using particle filter to do the location

estimation based on this model. In Section 4.3, simulation in MATLAB is conducted to prove the feasibility of the proposed localization algorithm, and provide us with more insights of the algorithm. The iOS implementation, named WiFi iLocate, and its performance in real situation, are presented in Section 4.5. Finally we present some concluding remarks in Section 4.5.

4.2 Localization Algorithm Design

The probabilistic location fingerprinting method uses Bayesian filtering to determine the location under estimation. Let $p(x_i \mid z)$ denotes the probability that the WiFi enabled device is in location x_i given the received signal vector is z. We select a location x_i if $p(x_i \mid z) > p(x_j \mid z)$, for i, j = 1, 2, 3, ... n, $i \neq j$.

Also let's assume that $p(x_i)$ is the probability that the smartphone is in location x_i, $p(z \mid x_i)$ is the probability that the signal vector z is received, given that the device is located in location x_i. The given decision rule is based on posterior probability

$$p(x_i \mid z) = \frac{p(z \mid x_i)p(x_i)}{p(z)} \tag{4}$$

Here we can assume that $p(z)$ is a constant since it is independent of x_i, so that the formula can be rewritten as

$$p(x_i \mid z) \propto p(z \mid x_i)p(x_i) \tag{5}$$

The estimated location x is the one obtains the maximum value of the probability

$$x = \arg\max_{x_i}[p(x_i \mid z)] \tag{6}$$

Traditionally, we collect the location fingerprints on the offline training phase and create a dataset to store them. During the online localization phase, we calculate the likelihood $p(z \mid x_i)$ of each location candidate based on the observed signal strength, and the estimated location can be decided by the Bayesian decision rule discussed above.

However, we can only collect fingerprints in discrete location, which makes this technique only applicable for discrete location estimation. On the other hand, smartphone user can be at any location, and move in a continuous manner. Therefore, we need to interpolate through the collected fingerprints. Gaussian process provides us such an advanced interpolation method.

4.2.1 Gaussian Process Regression for WiFi Signal Strength Modeling

A Gaussian processes essentially estimates a posterior probability distribution over functions from training data. The details on GPs can be found in [67]. We will give a brief introduction here.

Let's first define a function $f(x_*)$ be the posterior distribution that makes prediction for an arbitrary input x_*. And we have $D = \{(x_i, y_i) \mid i = 1, \ldots, n\}$, which is a set of training samples consists of n observations drawing from a noisy process $y_i = f(x_i + \varepsilon)$, where each x_i is an input sample in $\Re^d$ and each y_i is a target value in $\Re$, ε is additive Gaussian noise with zero mean and variance σ_n^2. For notational convenience, the inputs of the training set are grouped into a $d \times n$ matrix $\mathbf{X}$, and the observations y_i are grouped into a vector $\mathbf{y}$.

To estimate the posterior distribution over function $f(x_*)$ from training dataset D, GPs depend on a covariance function kernel $k(x_p, x_q)$, which specifies how the values at different points are correlated to each other. This kernel can be specified as any arbitrary covariance function, and we have chosen the widely used squared exponential kernel

$$k(x_p, x_q) = \sigma_f^2 \exp(-\frac{1}{2l^2} | x_p - x_q |^2) \tag{7}$$

Here, the hyperparameters σ_f^2 and l are the signal variance and the length scale which determine the strength of the correlation between different points.

Since we only have access to the noisy observations $\mathbf{y}$ instead of the true function value $f(x)$, we must add a term to account for observation noise in the covariance function:

$$\text{cov}(y_p, y_q) = k(x_p, x_q) + \sigma_n^2 \delta_{pq} \tag{8}$$

Here σ_n^2 is the Gaussian observation noise and δ_{pq} is one if $p = q$ and zero otherwise. For an entire set of input values $\mathbf{X}$, the covariance over the corresponding observations $\mathbf{y}$ can be written as

$$\text{cov}(\mathbf{y}) = \mathbf{K} + \sigma_n^2 \mathbf{I}, \tag{9}$$

where $\mathbf{K}$ is the $n \times n$ covariance matrix of the input values, defined as $\mathbf{K}[p,q] = k(x_p, x_q)$.

Note that the covariance between the observations is written as a function of the inputs, emphasizing the non-parametric nature of Gaussian process regression.

Now we can generate the posterior distribution over functions $p(f(x_*) \mid x_*, \mathbf{X}, \mathbf{y}) \sim N(\mu_{x_*}, \sigma_{x_*}^2)$ to estimate the function value for any arbitrary points x_*, given the training data $\mathbf{X}$ and $\mathbf{y}$:

The estimator's mean and variance are:

$$\mu_{x_*} = k_*^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \tag{10}$$

$$\sigma_{x_*}^2 = k(x_*, x_*) - k_*^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k_* \tag{11}$$

The hyperparameters σ_f^2, σ_n^2 and l control the smoothness of the functions estimated by a GP and can be learned from training data, by maximizing the log marginal likelihood of the observations conditioned on the hyperparameters. This learning process is completed offline right after the training dataset is built.

Gaussian processes (GPs) offer many advantages that make them suited for a localization system that utilizes WiFi signal strength:

Firstly, GPs are non-parametric, a mathematic model that can correctly fit the data is not required. Because GPs place a prior over the distribution of functions, many highly non-linear models can emerge from GP regression. Here we use GPs to approximately fit the non-linear WiFi signal propagation model.

Secondly, GPs do not require a discretized representation of an environment, or the collection of calibration data at pre-specified locations. They can predict signal strength measurements at arbitrary locations.

Thirdly, GPs provide uncertainty estimation for predictions at any given locations. This uncertainty is measured in variance, which takes into account the training data density and the noise of the data.

To apply GP in WiFi signal strength modeling, the input values $\mathbf{X}$ correspond to locations, and the observations $\mathbf{y}$ correspond to signal strength measurements gathered at these locations. The GP posterior is estimated from a collection of signal strength measurements corresponded with their locations. Assuming independence between different APs, we estimate a GP for each AP separately. After the generation of WiFi signal strength model, we can move to discuss the online particle filter based location tracking.

4.2.2 Location Estimation Based on Particle Filter

After the offline training phase to generate the GP based WiFi signal strength model, we are ready to use it in online localization.

During the online localization phase, we are going to determine the smartphone location using Bayesian filtering technique, implemented through a particle filter. Particle filter is able to handle any arbitrary probability density function. Previously, it has been adopted by several researchers in location estimation problem and has showed its advantages [66].

As discussed before, the Bayesian filtering is based on formula (4).

Here $p(z \mid x_i)$ and $p(x_i)$ represent a measurement likelihood model and a motion model, respectively.

The measurement likelihood model can be calculated using the posterior distribution of the signal strength at each location determined by the GP

$$p(z \mid x_*) \propto \frac{1}{\sqrt{2\pi\sigma_{x_*}^2}} \exp\left(-\frac{(z-\mu_{x_*})^2}{2\sigma_{x_*}^2}\right), \tag{12}$$

where μ_{x_*} and $\sigma^2_{x_*}$ are the posterior mean and variance at an arbitrary location x_*.

Location estimation algorithm using particle filter is performed according to the following steps:

(1) Particle Initialization

In order to realize location estimation in real-time using a smartphone, the computational complexity needs to be restrained. For particle filter, the computational complexity depends on the number of particles, N particles takes $O(N)$ time. We have tested different number of particles in our simulation to see how this number will affect the performance.

The initial location is calculated through weighted K nearest neighbor (W-KNN) method. It searches for K closest matches of known locations in signal space from the offline-built dataset. By averaging these K location candidates with adopting the distances in signal space as weights, the initial estimated location is acquired. This initial location estimation is used as the starting point for particles. We assume that the accuracy of the initial guess will not affect the localization performance. Larger initial error will only cause longer time for particles to converge to the actual location of the smartphone.

(2) Particle Movement

Next, the particles' coordinates are updated as each particle move. We choose not to use motion information under the assumption that for our WiFi localization system, this information may not be available or too noisy to be used. Therefore, in order to keep our location estimation only based on WiFi signals, the motion model is replaced with

random particle movement. That is, in every time step, the particle cloud is spreading in all directions and can reach random distance within a reasonable range.

(3) Weight updating

When the particles start moving, the collected WiFi received signal strength (RSS) continually changes the likelihood of all particles. For each particle, the predicted signal strength mean and variance are calculated from the GP model. They are used to calculate the corresponding particle's likelihood $p(z \mid x_i)$.

In every update, the likelihoods are calculated for each AP. Here we assume that the APs are independent of each other, so that we compute the likelihood of a complete set of readings from all the APs by multiplying the individual reading likelihoods together. This combined likelihood is then treated as the weight for particle. When all the particles' weights have been calculated, normalization is performed so that the sum of all the particle weights equals one.

(4) Resampling and location estimation

After particle weights are updated, we perform importance resampling to update the particles' location. In resampling, the weight of each particle is treated as a probability where this particular particle is chosen to be the estimated location. Those particles with higher weights will be picked more frequently than others. This is how the resampling is able to eliminate those wrongly moved particles and correctly track the smartphone's location. After the resampling process, the estimated location is calculated as the mean of all the resampled particles' location.

4.3 Simulations and Results Analysis

We have validated the proposed WiFi based localization algorithm through simulation. The simulation and result analysis is performed in MATLAB.

Wireless InSite, an EM solver by REMCOM, has been used to simulate a 40m by 40 m empty room with 4 APs, as shown in Figure 2. Mesh grid with grid size of 0.2 m is used to sample the WiFi RSS. A total of 40000 RSS data points are collected for each AP. We pick 400 equally spaced points out of the total as survey points to train the GP model. After the offline training process, a random path is generated in the room to see if the online localization algorithm could catch the actual path, and estimate the error between the true path and the estimated path.

We import all the WiFi RSS data into MATLAB, and demonstrate the online localization in a small animation in MATLAB. As shown in Figure 3, this true path, shown in red circle, starts at coordinate (4.5, 4.5) and ends at coordinate (25.5, 10.5). For simplicity, the path is on the grid with equal gap between each step. On the right, it shows a one-time simulation result. The particle distribution corresponds to different locations are shown in blue dots and the estimated path is shown in red stars. We have set the particle number to 1000 and use all 4 APs in this simulation.

To evaluate the accuracy of the localization algorithm, the simulation has been performed 100 times and the estimated location is compared to the ground truth location. The error is measured as the Euclidean distance between the actual location shown in red circle and the estimated location shown in red star. The average error in Figure 3 is 3.8 m.

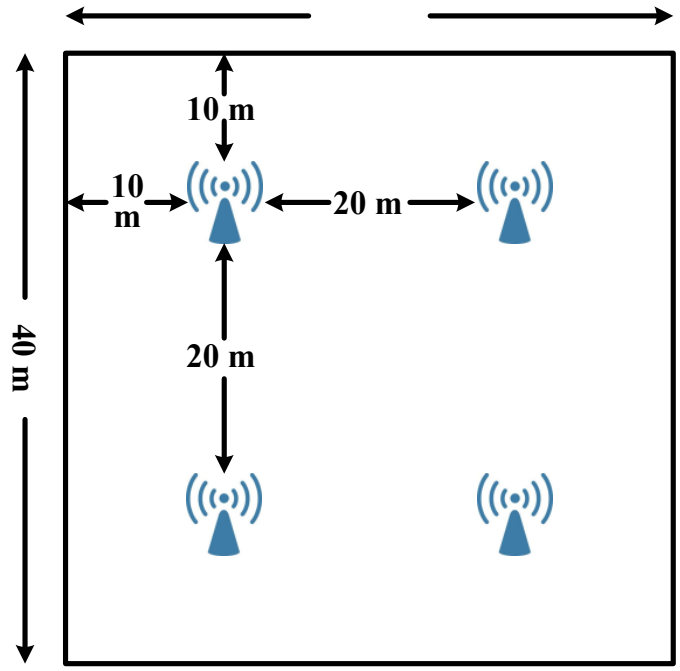


Figure 2: Simulated environment

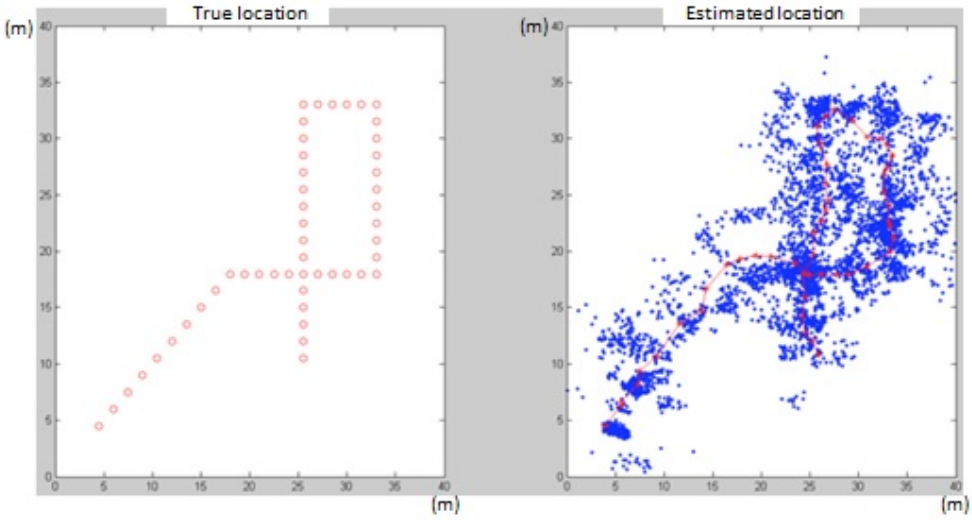


Figure 3: Particle filter estimation of a random path

Furthermore, we analyze the system performance under various situations. Firstly, we explore how the number of AP can affect the location accuracy estimation. We test the AP number from 1 to 4, and fix particle number to 1000. The result is shown in Table 1.

As can be seen in Table 1, when the number of AP increases, the error drops significantly. This is reasonable as more APs provide more information and can achieve better location estimation accuracy.

But the root of this performance deterioration lies in the fact of WiFi RSS fluctuation. If we have sparse deployment of AP, it's highly possible that two faraway locations happen to share similar WiFi RSS. Figure 4 illustrates this situation in the simulation. As we can see from the figure, there are two places with much higher weight than other places, if particle reaches either places, they will survive the resampling process, as shown in Figure 5. Thus, large error occurs when we calculate the estimated location as the mean of particle distribution. Liu et al [64] also demonstrate this root cause of large error for WiFi based localization in real indoor environment.

Table 1

The average error for different number of AP

Number of AP	1	2	3	4
Average error	14.8 m	7.4 m	6.5 m	3.8 m

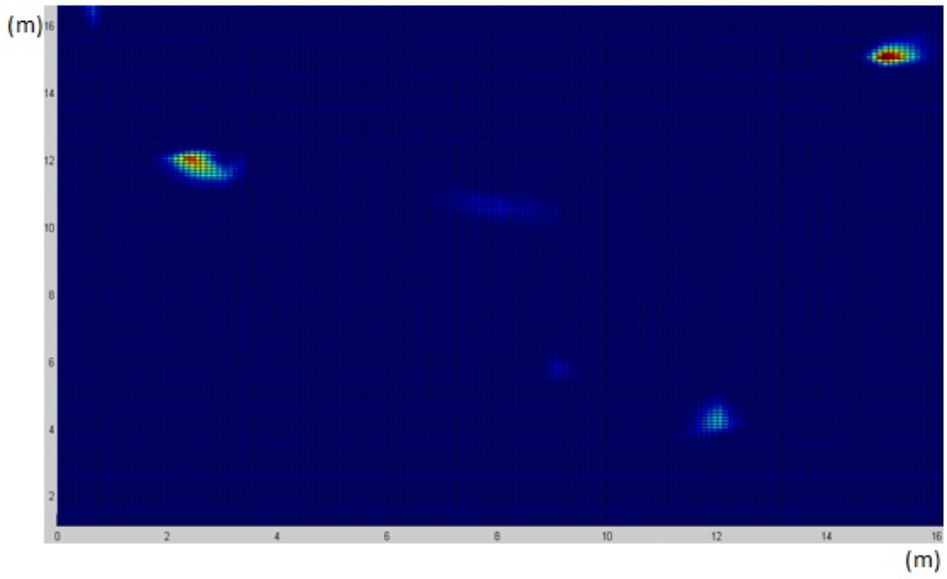


Figure 4: The situation when large error occurs

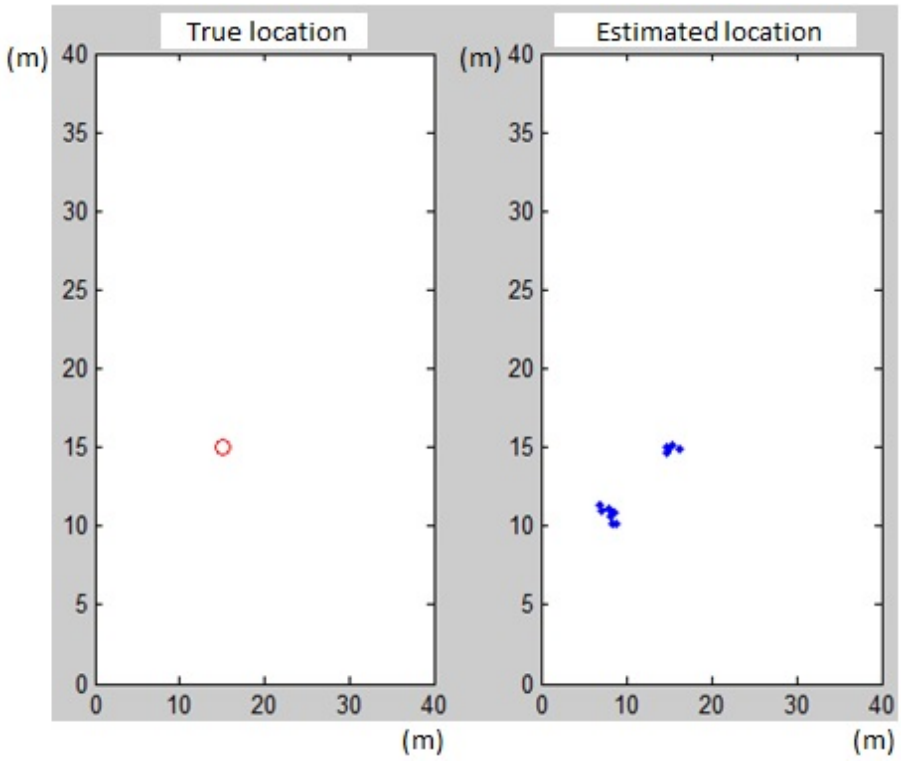


Figure 5: Particle distribution after resampling.

Secondly, we investigate how the initial accuracy will affect the localization performance. We test the initial error from 0.5 m to 3 m, with particle number and AP number set to 1000 and 4, respectively. The result is shown in Figure 6.

Figure 6 shows that the algorithm is not sensitive to the accuracy of the initialization. Better initial accuracy only gives a better performance in the beginning few steps. After 8 steps, the particle filter compensates the difference and gives a similar performance onwards. This result verifies our hypothesis in the particle initialization step, that particle initialization will not affect the location estimation performance as they will finally converge to the ground truth location.

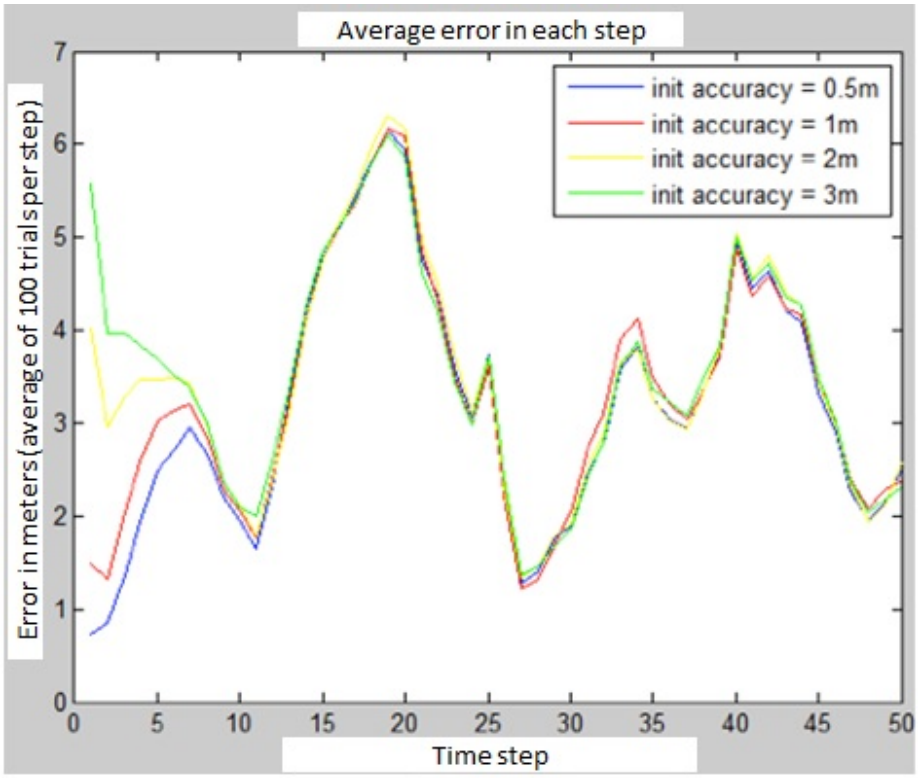


Figure 6: Localization performance in different initial error.

Thirdly, it is desirable to know how many particles are needed for an accurate localization. The number of particles from 200 to 2000 is tested, with AP number equals 4. The result is shown in Table 2.

As seen from Table 2, by increasing the number of particles, we can improve the localization performance. But using more particles means higher computational cost at the online localization phase. The benefit of accuracy improvement is very little when the particle number reaches a large enough value, as this simulation unveils, around 1000. This is a reasonable particle number to obtain good localization accuracy and real time response.

4.4 Implementation on iOS Platform

We realize the WiFi indoor localization algorithm on the iOS platform and built an app called WiFi iLocate. The system performance has been tested on the 3rd floor in Oakland University library.

Table 2

The average error for different number of particles

Number of particles	200	400	1000	2000
Average Error	4.6 m	4.2 m	3.8 m	3.7 m

4.4.1 System Architecture

Figure 7 presents the workflow of WiFi iLocate. In brief, we first import the floor plan into the system. With the floor plan display on the screen, we can set the survey points and scan WiFi. The scanned RSS values and corresponding BSSIDs are stored in an offline training dataset. After the construction of WiFi RSS dataset and preprocessing, we are able to perform the online localization by pressing the "Locate" button. Both the offline and online phases are completed on the iOS platform.

One important issue for WiFi scanning is that APs may be missed in a scanning cycle both during the offline surveying and the online localization. To overcome this issue, we perform multiple scanning in offline WiFi surveying and apply GP modeling to

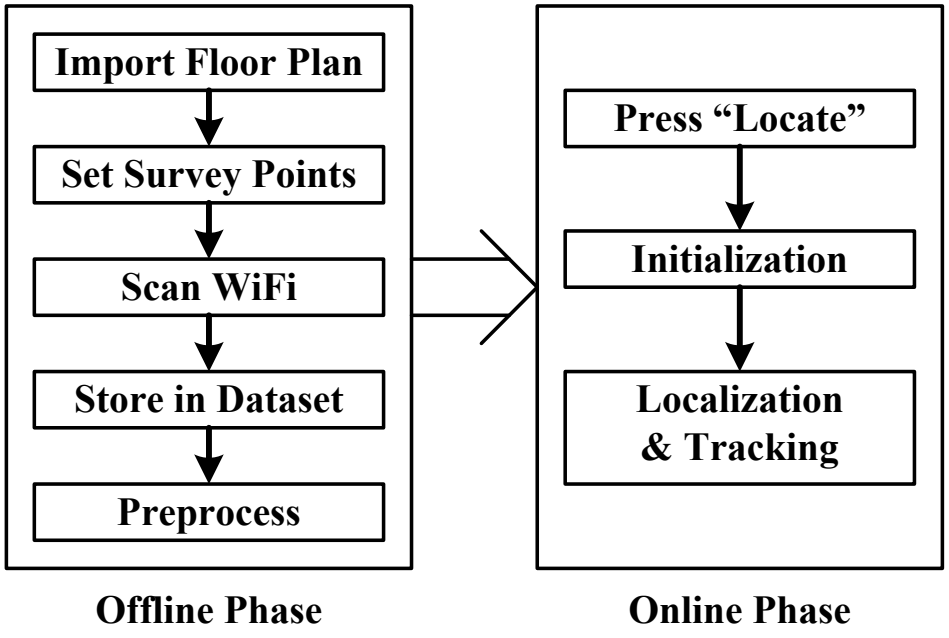


Figure 7: WiFi iLocate workflow.

deal with the missing value. During the online localization, interval is set as 4 seconds, which means we update our location estimation every 4 seconds. We scan once every period, compare the scanned BSSIDs with pre-store BSSIDs in the dataset, and use all the detected APs to calculate the current particles' weights.

The most computational costly step to generate the posterior of GP lies in the inverse of the covariance matrix: $(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}$, which takes time $O(N^3)$, where N is the number of survey points. Fortunately, this computation can be done before the real time localization step. After the creation of the training dataset, preprocess is performed and this inverse covariance matrix is stored in memory beforehand.

Right after we press the 'Locate' button, the iOS device starts scanning the WiFi RSS from all the APs it can detect. Particles are initialized through weighted KNN method. The number of particles is set to 1000 according to the simulation result. After particles are generated from the initial location, they start to move randomly, every 4 seconds the particles are resampled, as discussed in Section 4.2. Through particle filtering, we can locate the user and track the user movement in real time.

4.4.2 Experimental Evaluation

Our test environment is on the 3rd floor of Oakland University library, with seven APs installed. In total 18 survey points are collected for training purposes, as shown in Figure 8. Localization tests were conducted 10 times on a predefined path. The average length of the path is about 85m. During the traverse on the path, we measured the error distance between the estimated location and the ground truth location. The ground truth location is based on manual annotation of waypoints.

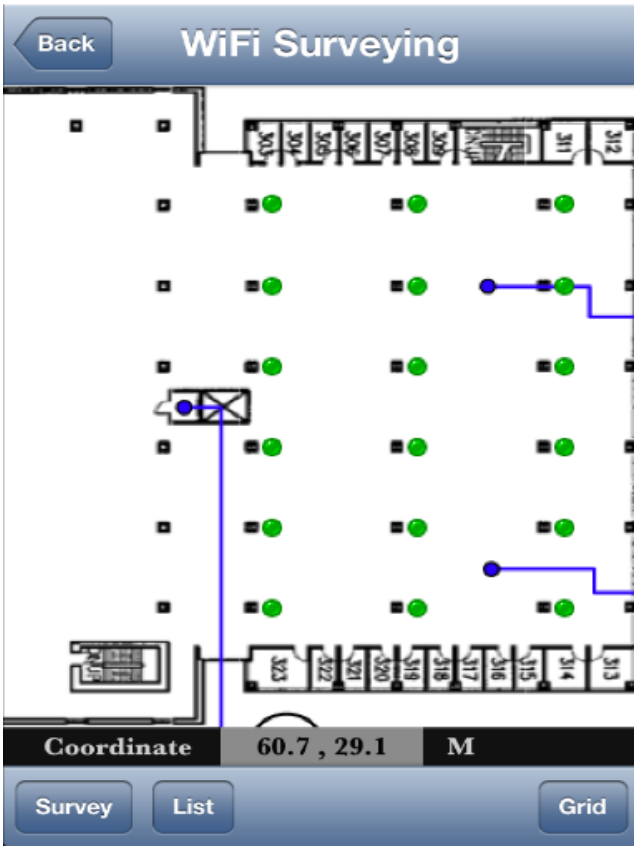


Figure 8: Survey points display on screen.

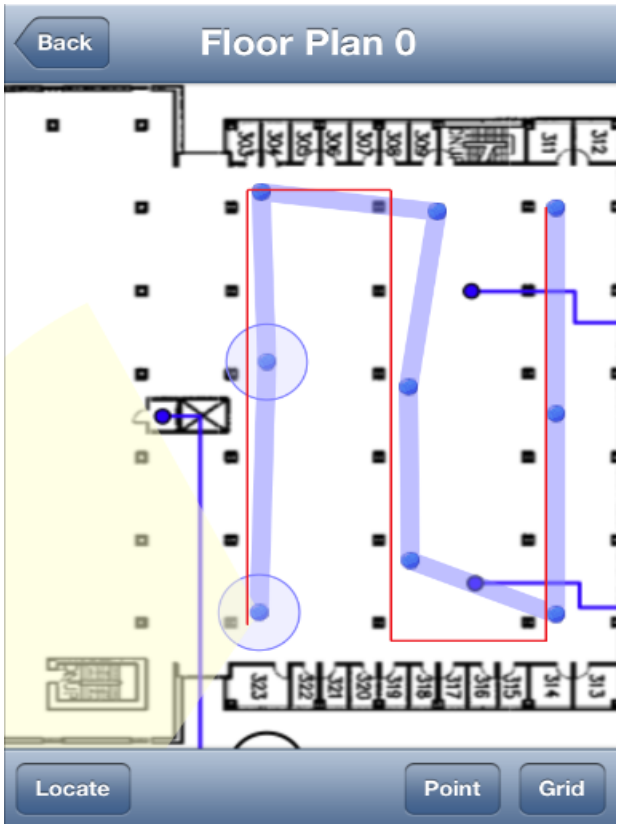


Figure 9: Estimated path vs. true path.

Comparison between the estimated path and the ground truth path is shown in Figure 9. The red line represents the ground truth path, while the blue dot and the thick stroke illustrate the estimated path.

The maximum error distance is about 5 m in the corner when we were making a turn. The mean and median error distances are 3.6 m and 2.9 m, respectively. Table 3 shows the advantages of our system when compare to pure W-KNN method. This real time test result clearly demonstrates the high accuracy of our WiFi based indoor localization system.

4.5 Conclusion

In this chapter, we have demonstrated an indoor localization system based on Gaussian process modeling of WiFi RSS dataset and particle filter localizer. The simulation result showed that our algorithm yields promising location estimation. The implementation on iOS platform and the test in real world situation proved that WiFi iLocate is a reliable, real-time, press-to-go indoor localization system. To the best of our

Table 3

Localization accuracy: WiFi iLocate vs. W-KNN.

Average error	Mean	Median	Maximum
WiFi iLocate	3.6 m	2.9 m	5 m
W-KNN	6.5 m	6.4 m	>10 m

knowledge, WiFi iLocate is the first app delivering such accurate, highly integrated indoor localization system on smartphone. Based on our system, many location-aware applications will be able to function properly indoor, provide more convenient service to people's daily life.

Although we have applied Gaussian process to reduce the labor work of collecting WiFi survey point in the offline training phase, this job remains heavy in a large indoor environment. Researchers have tried to overcome this problem using simultaneous localization and mapping (SLAM) methods [68]. These techniques can be tailored and combined into our system.

In the next chapter, we will combine motion sensors on the smartphone, in hope that multimodality could provide more location-related information and help us develop a dynamic motion model to improve the localization performance.

CHAPTER FIVE

WIFI BASED INDOOR LOCALIZATION WITH DYNAMIC MOTION MODEL USING SMARTPHONE MOTION SENSORS

5.1 Introduction

Indoor localization is a key component to many location based services, such as patient monitoring in hospitals, tour guiding in museums or spot finding in parking lots. As GPS lacks the ability to function indoor, people are looking for other solutions to solve the indoor localization problem. In recent years, WiFi RSS based location fingerprinting technique are attracting more and more interest for indoor localization, as they can provide good accuracy with no modification to the infrastructure. But WiFi RSS has the problem of signal fluctuation due to the fact of multipath fading in indoor environment. To meet the uncertainty of signal fluctuation, many probabilistic methods have been proposed. The state-of-the-art is the Bayesian filtering technique, implemented as particle filter.

Particle filter consists of two components: a measurement likelihood model and a motion model. In [4], Ferris et al propose an algorithm to construct the measurement likelihood model using Gaussian process regression during the offline survey phase. In the online localization phase, they apply particle filter to determine the target location. In order to not only locate the target in one spot, but also track the target movement, they develop a motion model to update the particles' coordinates on the map. Limited by using only WiFi signal, they adopt a naive conditional probabilistic motion model, which does not unleash the total power of the particle filter.

As the smartphones are equipped with more and more different sensors, the future of localization and tracking system will most likely evolve towards systems that are able to fuse the information provided by multiple sensors in a mobile device. However, most of the existing localization systems nowadays either lack the ability to process the multisensory integration problem or rely on data collected from separated MEMS sensors and WiFi signals sniffed from a mobile handset [69]. The localization is performed by post-processing the data from different sensors on a laptop instead of real-time processing on a smartphone.

In last chapter, we adapt the Gaussian process modeling of WiFi RSS and particle filter based localizer to smartphone platform and develop an iOS app called WiFi iLocate. In this chapter, we investigate the motion sensors embedded in the smartphone. Currently, smartphones are equipped with various low cost motion sensors, such as accelerometer, gyroscope and magnetometer. The localization technique based on them is called pedestrian dead reckoning (PDR). However, due to the drifting nature of the motion sensors, PDR could only achieve limited accuracy and the localization error will accumulate in the long run. To overcome the problem, Li et al [70] applied particle filter to fuse PDR with indoor map information: In each particle propagation step, the algorithm checks whether the particles ended into obstacles or cross the walls. If they did, their weights are set to zero and get eliminated in the resampling step. This approach greatly improves the PDR based indoor localization performance but requires extra infrastructure information and detailed modeling of the indoor map.

Inspired by the previous study trying to combine the WiFi based indoor localization with PDR [71] [72]. We developed an dynamic motion model by fusing the

data provided by motion sensors embedded in smartphone. This motion model is incorporated with a Gaussian process regression based WiFi RSS model into the particle filter framework.

Specifically, we make the following contributions: By carefully examine the information provided by various sensors, we are able to fuse the data effectively, and process the multisensory integration problem in real time. To the best of our knowledge, our upgraded iOS app WiFi iLocate is the first one to achieve accurate, highly integrated indoor localization by seamlessly leveraging information from WiFi and motion sensors on smartphone.

This chapter continues as follows: In Section 5.2, we first give a brief overview of our previous work. Then we describe the dynamic motion model and how it can improve the particle filter based location estimator. In Section 5.3, simulation is conducted in MATLAB to provide with more insights of the algorithm. The upgraded iOS application and field tests are presented in Section 5.4.

5.2 Dynamic Motion Model Design

In pure WiFi based indoor localization, since we apply a simple random walk model to control the particle movement, we need a large amount of particle to reach sufficient accuracy, as the location estimation fully relies on the resampling step during the online localization. We are hoping that some of the particles will randomly hit the right spot and others will be eliminated. If we have enough particles, it is working for most of the time. However, there're cases when particles can be fooled to a wrong place. We will demonstrate this in Section 5.3. In order to fully solve the problem, we develop a dynamic motion model to better control the particle movement.

A motion model enables the prediction of the smartphone user's movement based on his current location. It is usually represented by the conditional probability $p(x_{t+1} \mid x_t)$, where x_t is current state of the particle and x_{t+1} is the next state. In order to track the user in real time, it is important to develop a motion model doing accurate estimation of his actual movement. We present three steps dynamic motion models with the help of motion sensors embedded in the smartphone.

(1) Stage transition detection

We define two stages for particles, standing and walking. Initially, all particles stay in the standing stage. We perform stage transition detection based on data collected from accelerometer.

When the user is standing still, it is expected that their mobile device will register little acceleration. Therefore, the standard deviation in the magnitude of acceleration $\sigma_{\|a\|}$ is selected to detect the stage transition. It tells whether the particle to stay or move. If $\sigma_{\|a\|} < 0.01$, it means that the user is standing still in a location, thus the particle's movement will be limited in a small circle with center in the previous location. If $\sigma_{\|a\|} \geq 0.01$, however, it is not sufficient to ascertain that the user is walking. For example, user hands' sudden movement could result in a larger acceleration. Thus, we exploit the repetitive nature of walking.

Figure 10 shows the acceleration data recorded by a smartphone carrying by a walking user. We can see that the acceleration data exhibits a highly repetitive pattern. This pattern arises due to the fact of rhythmic nature of walking. In order to determine

whether the user actually enter the walking state, we calculate the auto-correlation of the acceleration signal a for lag τ at the m^{th} sample as follows [73]:

$$\chi(m,\tau) = \frac{\sum_{k=0}^{k=\tau-1}[(a(m+k)-\mu(m,\tau))*(a(m+k+\tau)-\mu(m+\tau,\tau))]}{\tau*\sigma(m,\tau)*\sigma(m+\tau,\tau)} \quad (13)$$

where $\mu(k,\tau)$ and $\sigma(k,\tau)$ are the mean and standard deviation of the sequence of samples from $a(k)$ to $a(k+\tau-1)$.

If the user is walking, then the auto-correlation will spike the periodicity of the walker. We define $\psi(m)$ as the maximum of the auto-correlation between $\tau_{\min} = 30$ samples and $\tau_{\max} = 60$ samples with sampling frequency of 50 Hz. If both $\sigma_{\|a\|} \geq 0.01$ and $\psi(m) \geq 0.8$ are satisfied, then we set the state equals walking.

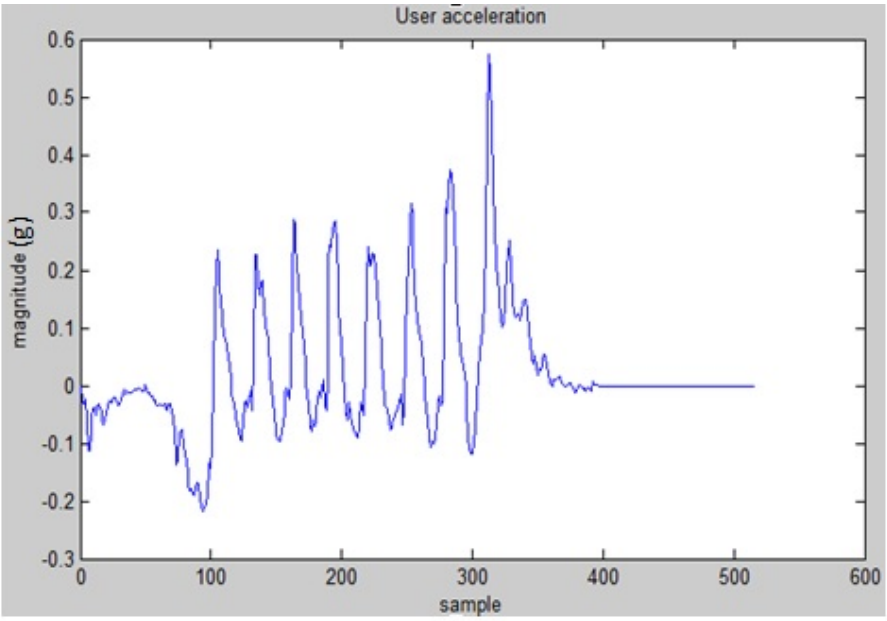


Figure 10: User acceleration during walking.

(2) Step counting and stride estimation

Once we have determined that state is walking, step counting and stride estimation are performed to calculate the walking distance of the user.

As shown in Figure 11, the step counting is realized by dividing the duration of sample when the maximum auto-correlation $\psi(m) \geq 0.8$ by τ_{opt}, and round up to an integer value. τ_{opt} is determined by simply finding the most frequently occurred τ in the duration when $\psi(m) \geq 0.8$.

Because human stride is not constant during walking, the stride size is determined by dynamically checking the acceleration sequence. We apply an empirical equation based on [74] to estimate the stride size.

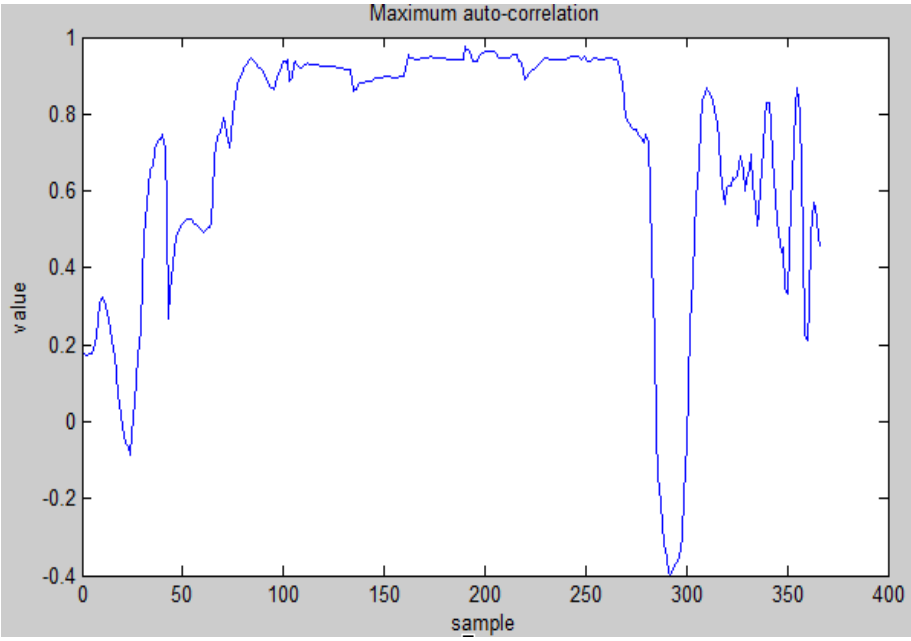


Figure 11: Maximum autocorrelation for step counting.

$$Stride = 0.98 * \sqrt[3]{\frac{\sum_{k=1}^{N} | a_k |}{N}} \quad (14)$$

where a_k means the measured acceleration and N represents the number of sample in one period of walking.

(3) Heading detection

The mobile phone's magnetometer provides heading orientation of the phone relative to the magnetic north. There are many researches about how to induce the user walking direction from the magnetometer reading with phone placing on different parts of the human body [71] [72]. For example, the phone may be placed in pants, bounded on arms or hold in hands. In our case, we use the phone to track our location in an indoor environment, with our current location and walking path display on the screen. Thus, we can simply assume that the phone will only be hold in hand and in portrait direction. This is a valid assumption for navigation application on smartphone and it reduces the complicated orientation induction problem to simple heading detection.

The iOS navigation API provides device heading information based on magnetometer reading. However, this heading direction is easy to be disturbed indoor. Therefore, we only use it as the initial heading. Further heading direction is calculated from device yaw attitude, which acquired from gyroscope. Figure 12 shows the device yaw attitude changing when user is walking.

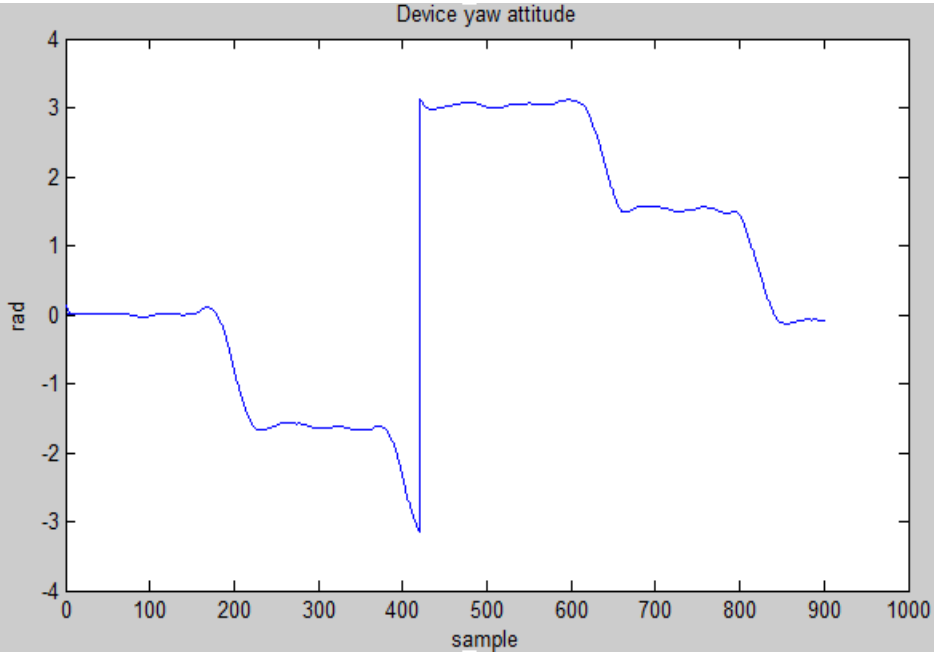


Figure 12: Device yaw attitude changing during walking.

To determine the new location, we first need to calculate the step number and stride length, and then estimate the movement as follows:

$$x(t+1) = x(t) + ((l(t) + \delta l(t)) * \cos(\theta(t) + \delta\theta(t)) \quad (15)$$

$$y(t+1) = y(t) + ((l(t) + \delta l(t)) * \sin(\theta(t) + \delta\theta(t)) \quad (16)$$

where $l(t)$ and $\theta(t)$ are the estimated step length and heading direction, while $\delta l(t)$ and $\delta\theta(t)$ are the zero mean Gaussian noises on the length and direction, respectively.

The motion dynamic model fuses the information provided by different motion sensors, and indicates a higher likelihood field in particle filtering. There are four scenarios depending on the access of different kinds of motion sensors. When an

accelerometer is available, the distance traveled can be estimated, based on step counting and stride length estimation. Otherwise, the distance is estimated with an empirical maximum speed, for example, 1 m/s. If a gyroscope and a magnetometer are available, the user heading is detected, if not, the heading remains unknown. Assuming an open space, the calculation of the likelihood field is shown in Figure 13. The grid points denote all possible state candidates for the next epoch, and the black dot is a state candidate of the current epoch. When only the distance traveled is measured, the likelihood field (shaded area) is located within a ring zone around the center, as shown in Figure 13(a). The radius and width of the ring are determined, based on the measured distance and its uncertainty, respectively. If only the user's heading is detected, and an empirical maximum speed is used to calculate a maximum walking range within a time interval, the likelihood field is shown in Figure 13(b). The angle of the shaded zone is determined based on the heading and its uncertainty. If both the distance and heading are measured, the likelihood field is shown in Figure 13(c). This is the case for our system, which greatly reduces the amount of particles needed for precise localization, thus decreasing the computational complexity in particle filtering. Finally, if we have no access to any motion sensors, assuming a maximum walking range, the likelihood field is located within a whole circular area, as shown in Figure 13(d).

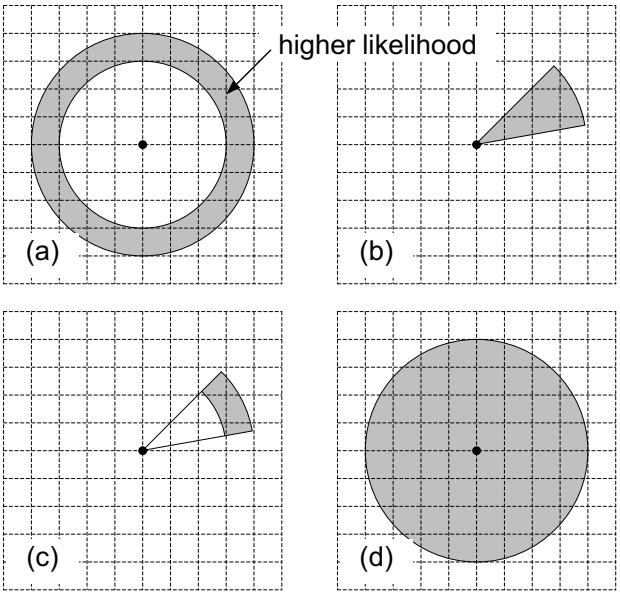


Figure 13: Motion dynamic model likelihood field.

5.3 Simulation of Performance Deterioration in WiFi Based Localization

We are interested in how the extra sensors, like accelerometer, gyroscope and compass could improve the localization accuracy, as our target devices - smartphones, do have these common sensors equipped.

A dynamic motion model was built based on the information provided by motion sensors to constraint the particles' movement. This motion model guides the particles to certain direction range and limits the step length to a reasonable range.

Figure 14 shows an estimated path with the help of motion model. We can observe that the particle distribution is more concentrated around the true path as compared to Figure 3.

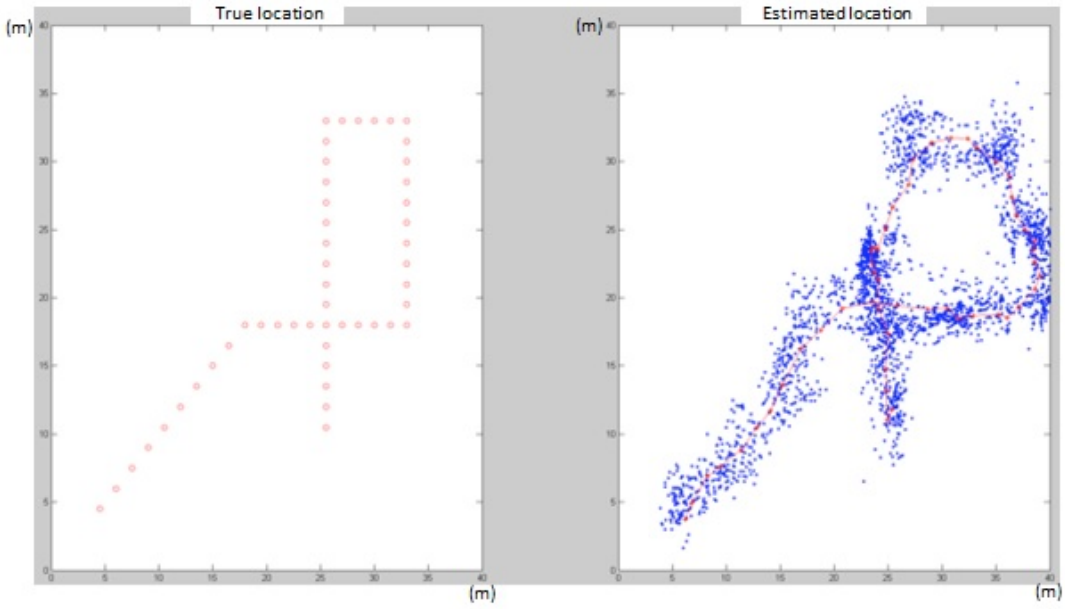


Figure 14: Particle filter location estimation with motion model.

Table 4 shows the comparison result of no motion model against with motion model under different number of AP. It can be seen that the motion model greatly increases the location estimation accuracy. When we don't have many APs to provide enough information of the WiFi signal strength, we will need to use the motion model to guide the particle movement. Even with enough AP, the performance still improves by about 1 m in location estimation accuracy with the motion model equipped.

By introducing a dynamic motion model, we are able to control the propagation of particles. Figure 15 shows the localization comparison in each time step between no motion model and with motion model in sparse AP situation. We can see that the motion information successfully helps solving the root cause of large error.

Table 4

Localization accuracy: No motion model vs. with motion model.

Average error	No motion model	With motion model
1 AP	14.8 m	5.5 m
4 APs	3.8 m	2.9 m

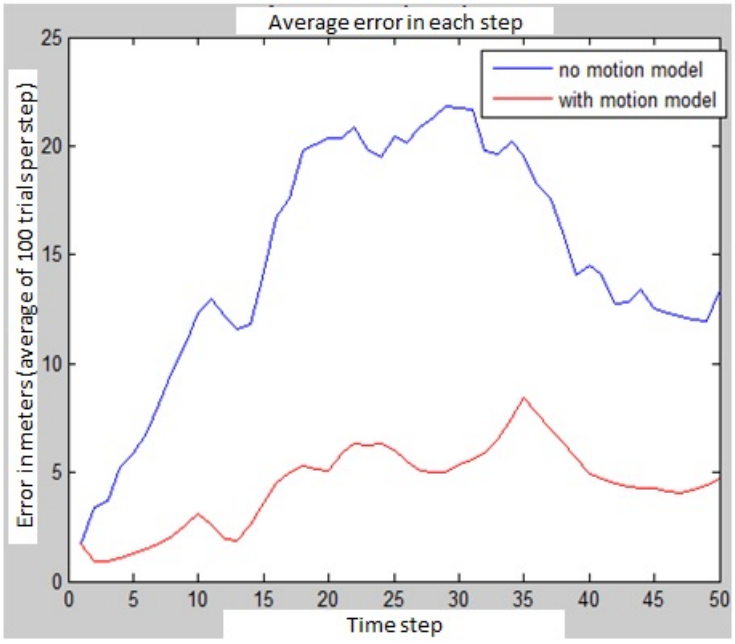


Figure 15: Localization error comparison, 1 AP situation

5.4 Implementation on iOS Platform

The dynamic motion model is implemented within the localization and tracking system. By introducing the iOS Motion API, we can record device motion data from accelerometer, gyroscope and magnetometer embedded in the smartphone in real time. These raw data related to the user acceleration, device attitude and heading direction are being logged simultaneously during the scanning of WiFi RSS. In each localization epoch, the motion information is processed by the dynamic motion model, and controlled the particle propagation.

Figure 16 compares the estimated paths with the ground truth path. The dark red line represents the ground truth path, while the blue dot and the orange stroke illustrate the estimated path.

Table 5 compares the localization performance between the pure PDR, previous WiFi iLocate system, and the updated one with dynamic motion model.

By seamlessly fusing the motion information into the system, we not only improve the localization performance, but also reduce the computational complexity, thus the system response time has decrease, resulting in a more robust, real time localization and tracking system.

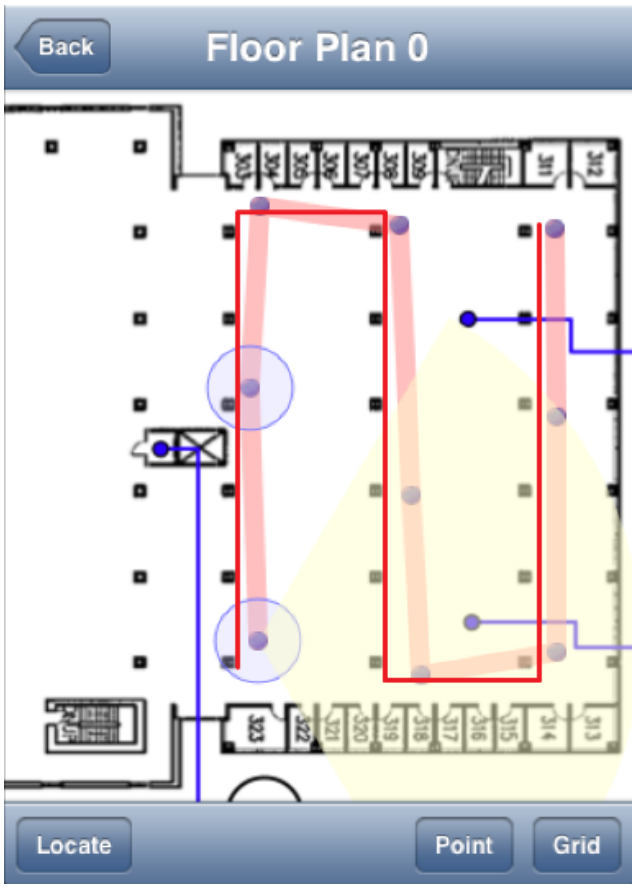


Figure 16: Comparison of the estimated path against the ground truth

Table 5

Localization accuracy comparison.

Average error	Mean	Median	Maximum
Pure PDR	9.5 m	8.8 m	>15 m
WiFi iLocate	3.6 m	2.9 m	5 m
WiFi iLocate with Motion Model	2.0 m	2.1 m	3.1 m

5.5 Conclusion

In this chapter, we have demonstrated an indoor localization and tracking system that is capable of integrating WiFi RSS and motion sensor information on smartphone. By introducing a dynamic motion model, we seamlessly fuse the motion information with the WiFi based localization technique. To the best of our knowledge, our updated WiFi iLocate is the first app delivering such accurate, highly integrated indoor localization system on smartphone platform.

As Google has announced in Project Tango [75], they have developed a prototype phone with powerful vision and 3D sensors. We believe the key technology for future localization lies in how to effectively fuse the information provided by various sensors. In the next chapter, we are looking to combine LiDAR and camera sensors into WiFi iLocate.

CHAPTER SIX

PORTABLE 3D VISUAL SENSOR BASED INDOOR LOCALIZATION

6.1 Introduction

Precise indoor localization is a key component in many location-aware applications, such as navigation, interactive gaming, and merchandise advertising. People have come up with various approaches for smart mobile device user localization in GPS-denied indoor environment. To name a few, radio frequency identification devices (RFID), wireless fidelity (WiFi) and pedestrian dead reckoning (PDR) are the three frequently used ones. However, the RFID based system requires a large amount of tags to be installed in the indoor environment before the user can locate his mobile device [1]. WiFi based localization techniques have the problem of signal fluctuation due to multipath fading effect in indoor environment. The PDR approach suffers from the fact that the inertial sensors on the mobile device are low cost micro electromechanical system (MEMS) sensors, which has relatively low accuracy. Thus the integration drift will cause the localization deviation to accumulate over time, and become unacceptable in the long run. Moreover, the above methods provide only location coordinates, and therefore have limits in further localization potential, such as obstacle detection, sign recognition, and disabled assistance, etc., which are very important features in location-aware services.

In order to achieve the above goals simultaneously, computer vision technologies are introduced to the localization task. A majority of the existing visual information based localization approaches use 2D images to localize the mobile device's user. The main

idea of 2D image-based localization is to first store the scene images, or features of these images, into a database, indexed and related to predefined geo-locations. A new query image to be localized is preprocessed with feature extraction and matched to the images in the database. The top ranking matched image is retrieved from the database. Since the retrieved image is geo-tagged, the localization task has become pose recovery between the matched images. Liang et al build an image based localization system following the above procedure [12]. The complete pipeline of their method from image retrieval (20,000 images in the database) to pose recovery takes 10-12 seconds to output a solution for a single query image on a 2.3GHz i5 laptop. The high computational cost makes it unpractical on mobile platform. Jaramillo et al develop a 6-Degree of Freedom (DoF) pose localization system using a monocular camera [76]. The database is constructed of a dense 3D point cloud, which can be projected to 2D to form a virtual view using the previous localized pose of the mobile device's camera. The 2D-3D point correspondences are obtained between the current captured image's 2D features and their matches on the virtual depth image (projected 3D points). This forms a perspective-n-point (PnP) problem, which can be solved for the relative transformation between the current camera pose and the virtual view. Their results show that a 2D camera can be localized in a 3D model in real time. The most time consuming part of this method is the estimation of initial pose, which can take hours if it's in a large indoor environment. To increase the initialization process, Ruiz et al apply multiple sensors to estimate a coarse initial location [19]. They divide the indoor 3D model into different sub areas and refine the initial estimation with SIFT feature matching.

Inspired by previous works, we develop an indoor localization algorithm on mobile platform. A portable 3D visual sensor is mounted onboard to do localization in a prebuilt 3D point cloud. During the offline training phase, we not only model the indoor environment as a 3D point cloud, but also apply Gaussian process (GP) regression to model the WiFi received signal strength (RSS) dataset, which will be used in initial pose estimation. Once we have the 3D model and GP model of the indoor environment, we are ready for online localization. After the coarse estimation of the initial pose using WiFi RSS data, the user location is narrowed down within a sub area of the 3D model. We further calculate the pose of the mobile device by matching features between online captured images with key frame images of the prebuilt 3D model. Then we can consecutively estimate the pose of the device by solving the rigid transform between online captured 3D point cloud and local 3D model using ICP algorithm [77]. Moreover, RANSAC algorithm [78] is applied to improve the 6-DoF pose estimation accuracy.

Specifically, we make the following contributions: We built an indoor localization system on iOS platform based on 3D visual sensor and WiFi RSS. To the best of our knowledge, our iOS app is the first application attempting to achieve 6-DoF pose estimation in an indoor environment modeled as a 3D point cloud and GP based WiFi RSS model. We believe this app will boost various location based services around it.

The rest of the chapter is organized as follows. Section 6.2 explains the details of our indoor localization system, including offline training phase and online localization phase. The experimental results in typical indoor environment are presented in Section 6.3. We conclude our work in Section 6.4 with a discussion of future improvement.

6.2 System Setup

This section describes the design of our proposed indoor localization method. Figure 17 presents an overview of the system workflow. During the offline training phase, color images, depth images and WiFi RSS are recorded in predefined survey points. By manually or automatically aligning the data in each survey point, we are able to generate a detailed 3D model of the indoor environment. The scanned WiFi RSS values and corresponding BSSIDs of the access points are used to train a GP model. Based on the survey point locations, we further divide the indoor environment into different sub areas. During the online localization phase, we first determine an initial coarse estimation that indicates a sub area where the mobile device user seems to be, using the WiFi RSS captured in the air. Next, making use of the image captured by the camera, we perform a matching process against the key frame images in the 3D sub model to determine the 6-DoF pose of the mobile device. Finally, we can continuously update the device pose by finding the 3D points correspondences between online captured 3D point cloud and local 3D model using ICP algorithm. In the following subsections, we will explain the details of the offline training phase and online localization phase.

6.2.1 3D Modeling of Indoor Environment

First of all, our localization method requires a detailed 3D point cloud with color and texture information. The color and texture information are usually captured by cameras, while the depth information are usually obtained using range sensors. In order to build a color 3D model, we need to fuse the color images with depth images

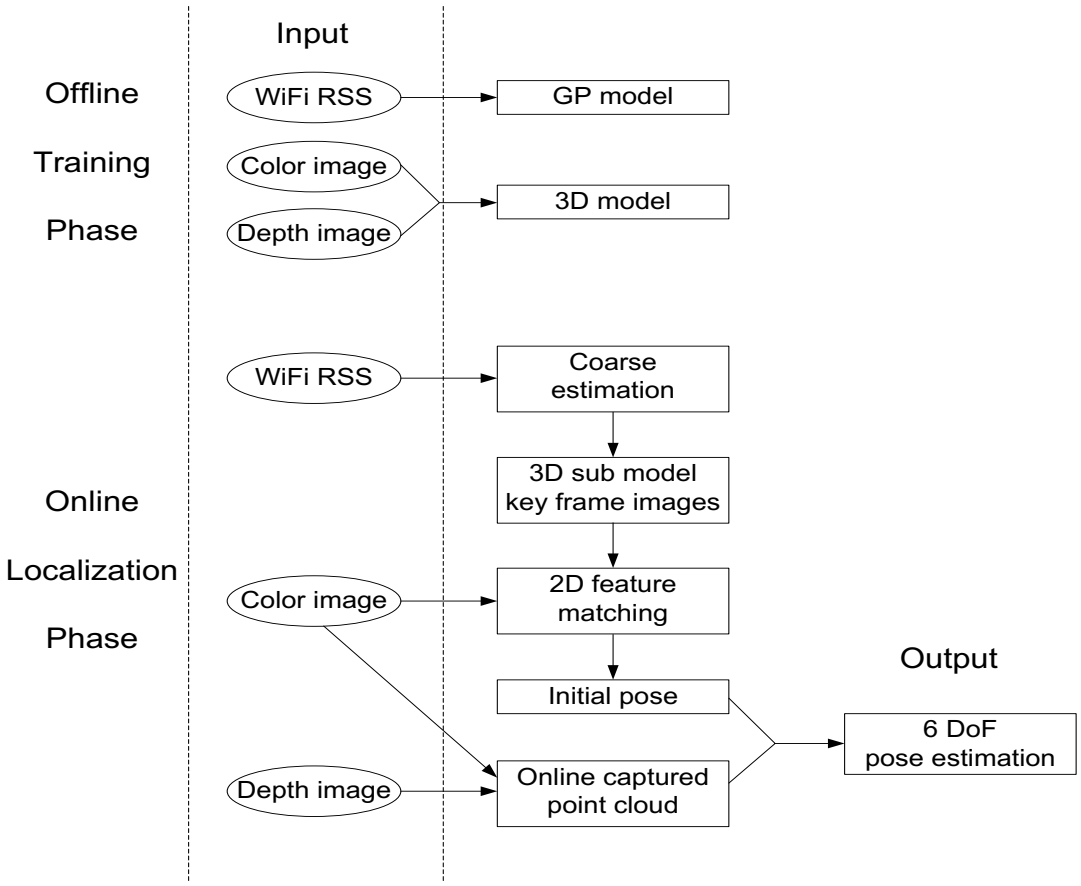


Figure 17: System workflow

effectively. There are mainly two methods to achieve this goal. One is to use RGB-D device such as Kinect or PrimeSense sensor. The RGB-D device is able to register the color images with the depth images. Then we can apply ICP algorithm to automatically align individual, consecutive local 3D point cloud to generate a complete 3D model. This approach is fast and easy to implement. The drawback for this approach is that the RGB-D sensor can only provide depth information up to a very limited range (around 5m). And its depth estimates are pretty noisy compare to LiDAR. Therefore, we use it to model a small indoor room area by placing it in the middle of the room and rotating around itself to scan the entire room. An example of a room model is shown in Figure 18.

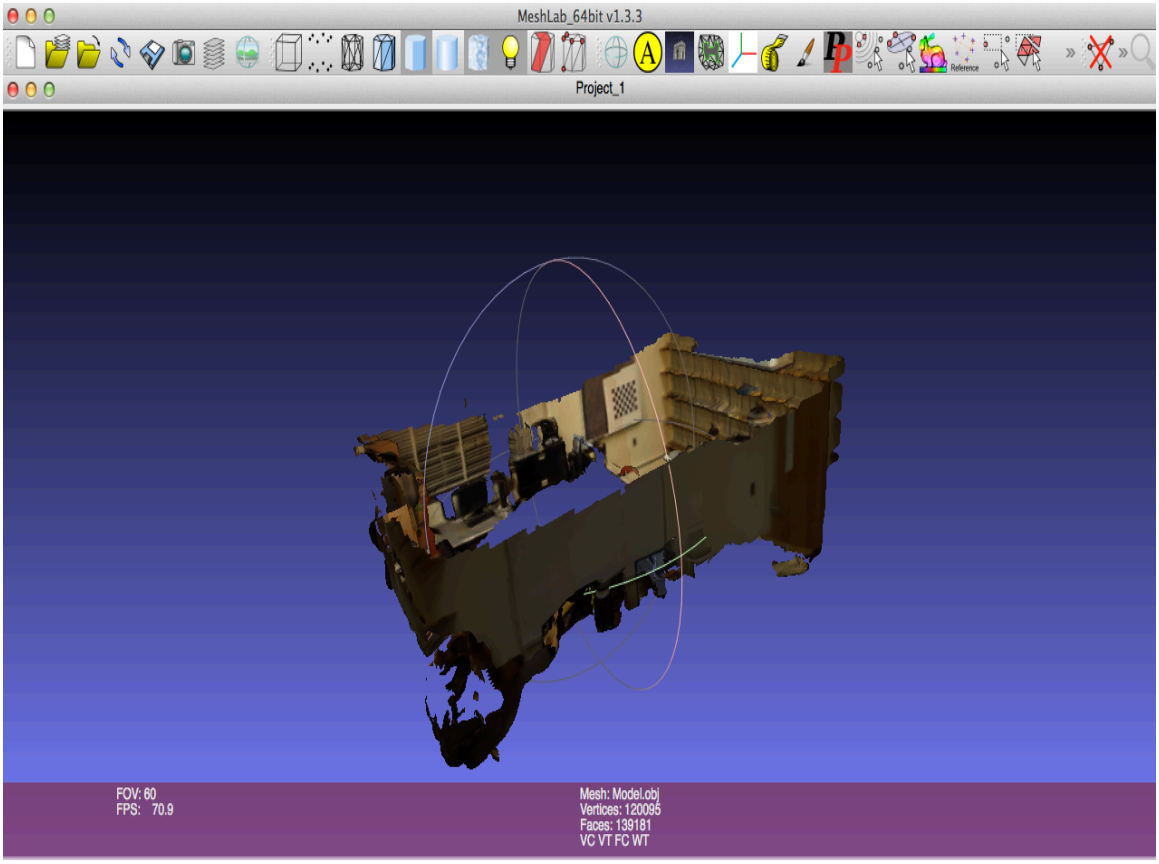


Figure 18: 3D model of a small room

In order to model a large indoor area such as corridor, we choose to fuse the data from camera and LiDAR. Figure 19 shows the assembly of our LiDAR-camera data acquisition system. It includes a 2D line-scan LiDAR, a servo, and a 2D digital camera. The LiDAR is installed on the servo and the camera is rigidly mounted on top of the LiDAR. The whole system is mounted on a pushcart for stop-and-go scanning.

Fiducial target-based extrinsic calibration [79] is applied to acquire transformation matrices between LiDAR and the camera. Based on the transformation matrix, we perform registration to fuse the color images from the camera with the 3D point cloud from the LiDAR.

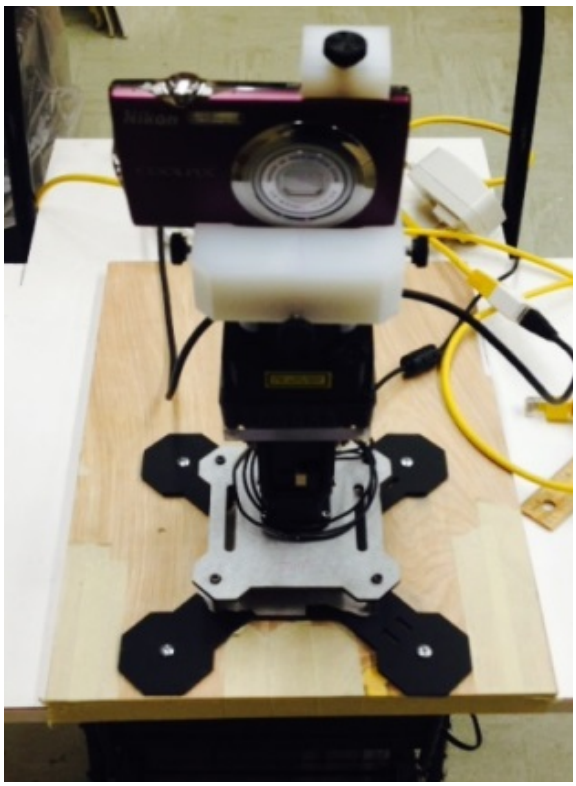


Figure 19: The LiDAR-camera scanning system

As shown in Figure 20, a 3D point in the LiDAR calibration plane is represented as $P_l = [x, y, z]^T$ and its related pixel in the camera image plane is described as $P_c = [X, Y, 1]^T$. The 3D point P_l with intensity information is projected to a calibration plane under a pinhole camera model. The calibration plane is defined at $z = f$ and the projected point in the calibration plane is shown as $P = [u, v, 1]^T$. Based on similar triangle rules, we have the following relationship:

$$u = f\frac{x}{z}; v = f\frac{y}{z} \tag{17}$$

where f is the focal length of the camera. In order to fuse the information from LiDAR and the camera, we need to look for the relationship to match P and P_c.

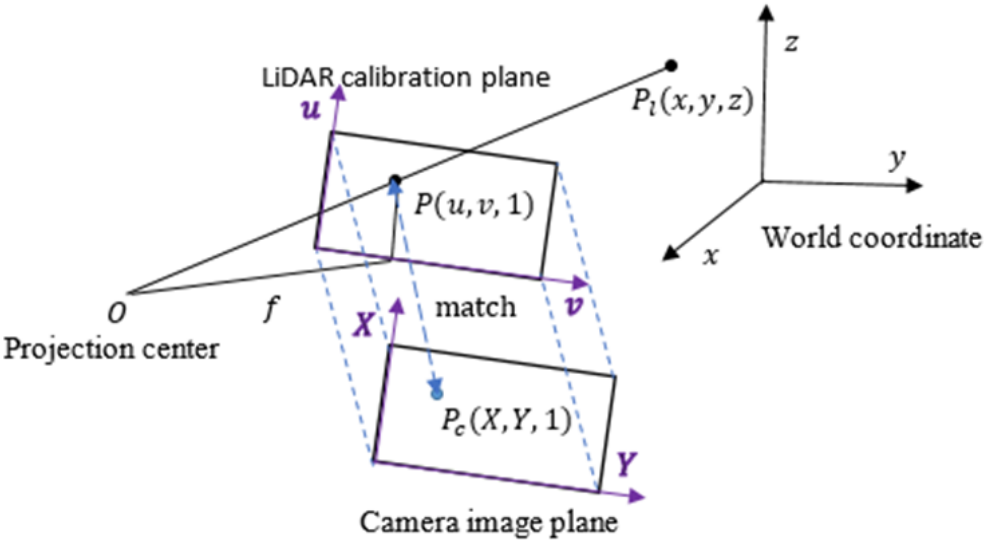


Figure 20: Pinhole camera model.

Figure 21 gives a workflow of the extrinsic calibration procedure. After projecting the 3D points to the calibration plane, we get a 2D point cloud. These 2D points are interpolated to generate a LiDAR intensity image. The problem of extrinsic calibration has become how to find the geometric constraints between a LiDAR intensity image and a camera image using the checkerboard pattern. The transformation of the checkerboard pattern from the LiDAR calibration coordinate frame to the camera coordinate frame is represented by a rigid 3 x 3 matrix T .

$$P_c = TP \tag{18}$$

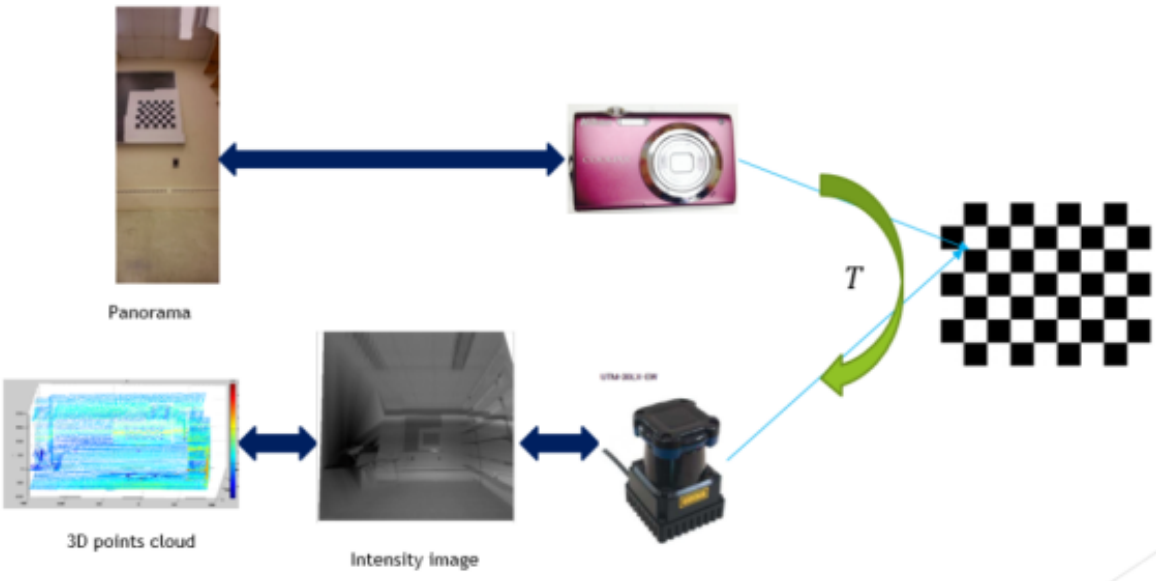


Figure 21: Extrinsic calibration procedure.

As shown in Figure 22, to obtain the features of a checkerboard accurately, we select a Region of Interest (ROI) from the LiDAR intensity image and the camera panorama for the checkerboard pattern. Next, we take advantage of Random Sample Consensus (RANSAC) algorithm to find the correspondences between the LiDAR intensity images and the camera panorama images. In RANSAC, a pair of points is selected as an inlier only when the distance between them falls within the specified threshold. The distance metric used in RANSAC is as follows:

$$D = \sum_{i=1}^{N} \min(d(P_c, T(P)), \xi) \tag{19}$$

where P is a point in the LiDAR intensity image, P_c is a point in the camera panorama image, $T(P)$ is the projection of a point on the intensity image based on the transformation matrix T, d is the distance between a pair of points, ξ is the threshold, and N is the number of points.

The algorithm for generating the transformation matrix is summarized below:

(1) Find the inliers for the corners of checkerboard based on RANSAC algorithm.

The RANSAC algorithm follows these steps:

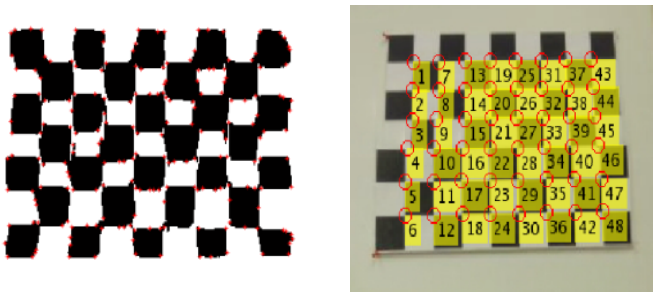
(a) Randomly select three pairs of points from the LiDAR intensity image and camera image to estimate a fitting model.

(b) Calculate the transformation matrix T from the selected points.

(c) Change the T value, if the distance matrix of a new T is less than the original one.

(2) Choose the transformation matrix T, which has the maximum inliers.

(3) Use all inlier point pairs to compute a refined transformation matrix T.

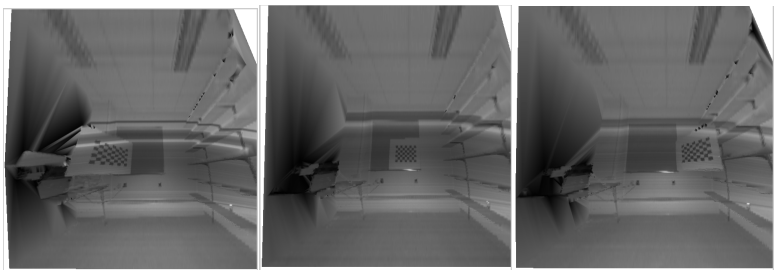


(a) (b)

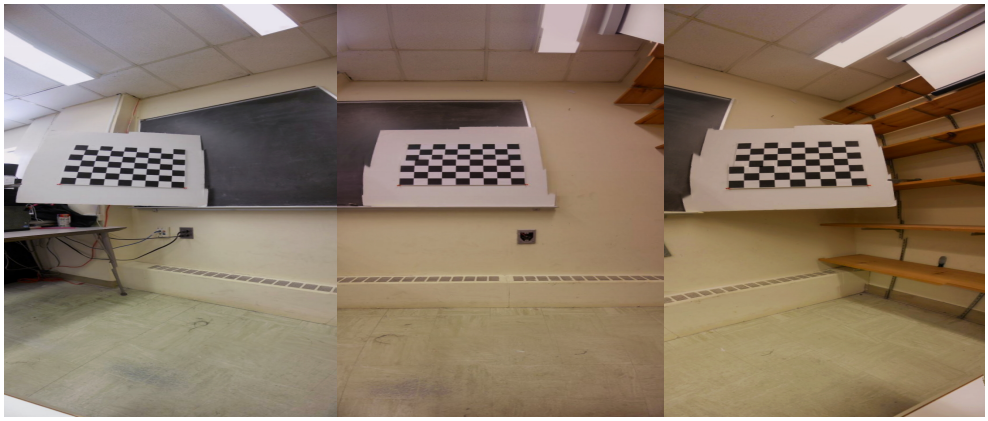
Figure 22: (a) Region of Interest (ROI) of LiDAR intensity image; (b) ROI of camera panorama.

We place a checkerboard in three different locations (left, center, and right) to generate camera panoramas and LiDAR intensity images separately. Based on these images, we derive the three transformation matrices. As shown in Figure 23, the color images are seamlessly fused with the intensity images. After generating the transformation matrices, we are able to stitch three camera panoramas together and fuse them with one LiDAR intensity image by applying these transformation matrices. The results are shown in Figure 24. After the fusion, we find the correspondences and assign color value to each LiDAR 2D point.

Finally, we back-project the textured 2D points to 3D color points cloud, as shown in Figure 25. Note that there are dark areas at the border, due to the fact that those points with distance from the projection center shorter than the focal length cannot be projected to the calibration plane.



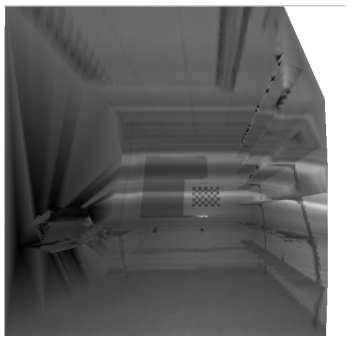
(a)



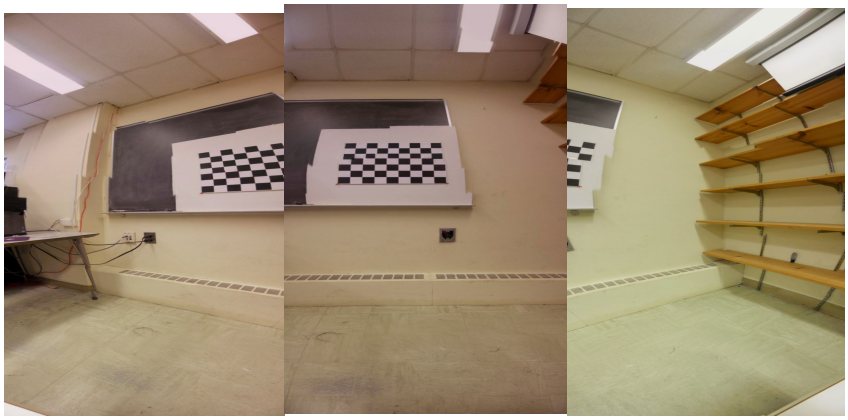
(b)

(c)

Figure 23: Transformation matrices derived process: (a) LiDAR intensity images; (b) Panoramas; (c) Stitched panoramas with intensity images.



(a)



(b)

(c)

Figure 24: Image stitch process: (a) LiDAR intensity image; (b) Panoramas; (c) Stitched three panoramas in one intensity image.

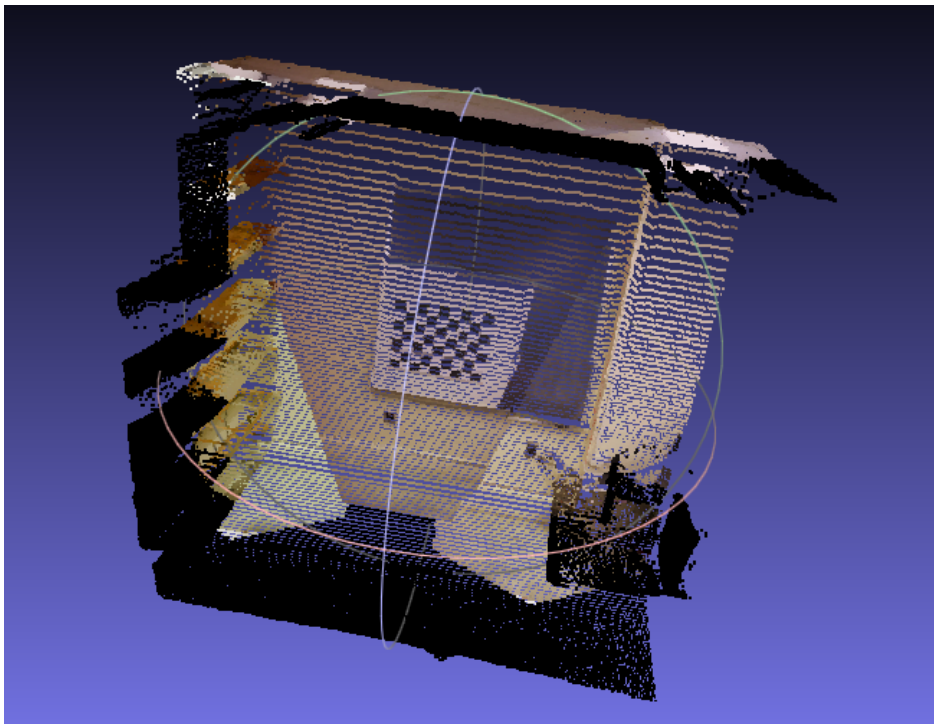


Figure 25: Color 3D points cloud.

The extrinsic calibration result is applied to a large indoor environment. The LiDAR-camera scan system is mounted on a pushcart in order to record the data in stop-and-go mode. By manually aligning the data in each survey point, we can get a detailed 3D model of the indoor environment. At the same time, the 3D model is partitioned, based on the survey point locations. Figure 26 shows a 2D map of a large corridor area and survey point locations. The corresponding 3D model is shown in Figure 27. In order to construct a 3D model of the corridor, with an area of 630,000 square feet, we have collected 1,029,974 data points; each point with a XYZ and RGB value. Based on the high accuracy of the laser beam, this model is much more accurate than the 3D model generated from a RGB-D sensor. Admittedly, it is computational heavy to process the

data in the offline training phase to build a detailed 3D color point cloud. However, during the online tracking phase, we only need to match the captured image with a local model instead of the entire one, which significantly reduces the cost.

6.2.2 6-DoF Indoor Localization Based on 3D Visual Sensor

During the initialization step, we scan the surrounding WiFi RSS values and corresponding BSSIDs. We compare them with the offline built GP model to coarsely determine the user location. This initial estimation tells which partition the user locates in the 3D model. We then perform SURF [15] feature matching between online captured image with key frame images of the 3D sub model. The result is sorted based on the matching distance in feature descriptor space. The nearest neighbor is selected as the closest key frame image to the captured image. Since we have the 3D coordinates of

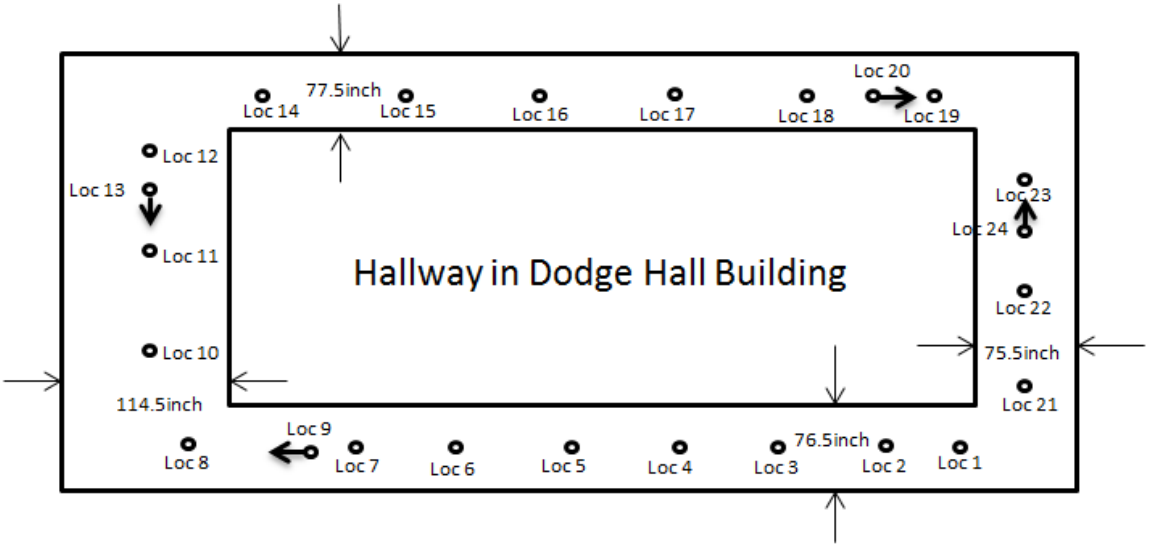


Figure 26: 2D map of a corridor.

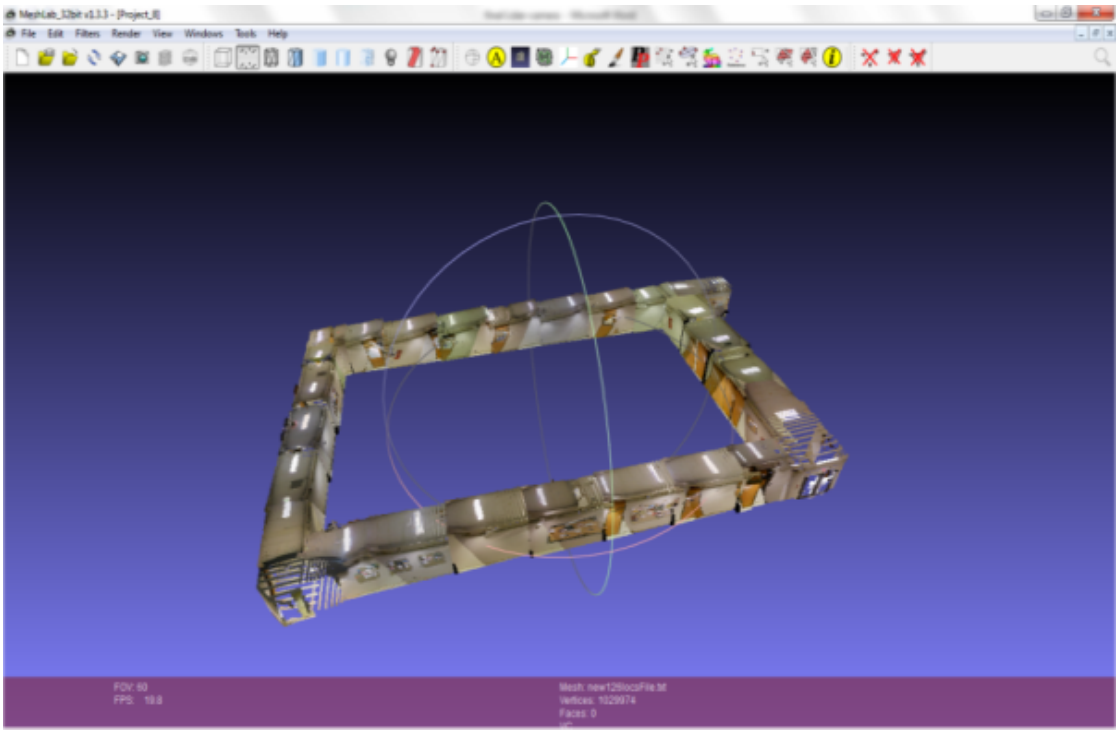


Figure 27: 3D model of a corridor.

the key frame image's feature points (given in its registered depth image), and the 2D coordinates of the captured image's corresponding feature points (specified in the image), these n feature points form a classic Perspective-n-Point (PnP) problem. By solving the PnP problem [80], we get the transformation matrix between the captured image and the key frame image. Therefore, we can calculate the initial 6-DoF pose of the user's mobile device $Pose_{t0}$ by simply multiply the global pose of the key frame image $Pose_{keyframe}$ with the transformation matrix $TForm_{t0}$.

Once the initial pose of the device is known, we perform pose estimation using the portable 3D visual sensor on the device. The procedure runs as follows: At a given

time step $t > 0$, we have an online captured 3D point cloud $OnlinePCloud_t$. Meanwhile, a local point cloud $LocalPCloud_{t-1}$ is generated from the previous pose $Pose_{t-1}$, computed at time $t-1$ using the prebuild 3D model. Both these point clouds $OnlinePCloud_t$ and $LocalPCloud_{t-1}$ have their own color image ($OnlineColorI_t$, $LocalColorI_{t-1}$) and depth image ($OnlineDepthI_t$, $LocalDepthI_{t-1}$). We perform SURF feature matching between $OnlineColorI_t$ and $LocalColorI_{t-1}$. The matched feature points' 3D locations are acquired from the corresponding depth images ($OnlineDepthI_t$, $LocalDepthI_{t-1}$). Thus, the pose estimation problem has become finding the rigid transform between two sets of 3D points, which can easily be solved with ICP algorithm. Figure 28 illustrates the pipeline of this process. Additionally, we employ RANSAC algorithm to improve the feature matching process. An example is given in Figure 29. We can see the RANSAC algorithm has removed many of the outliers.

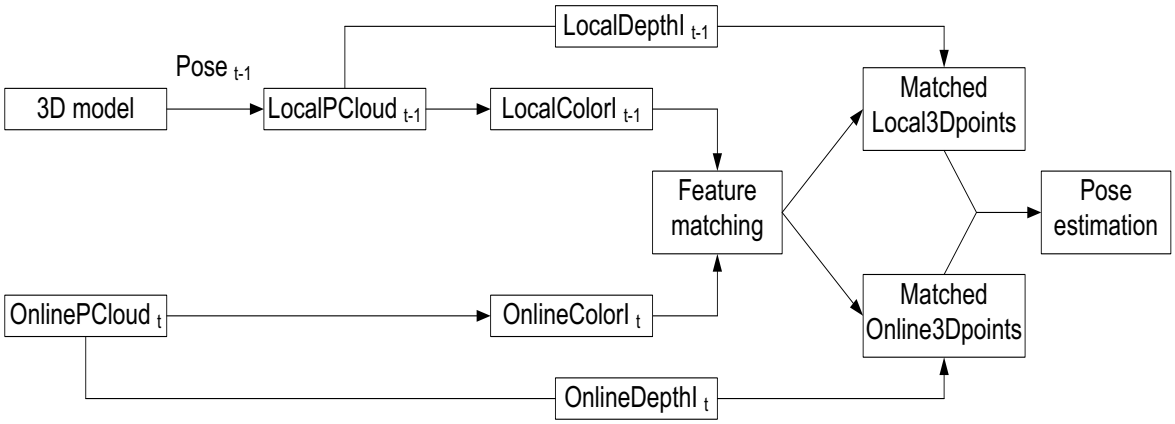


Figure 28: 6DoF pose estimation pipeline

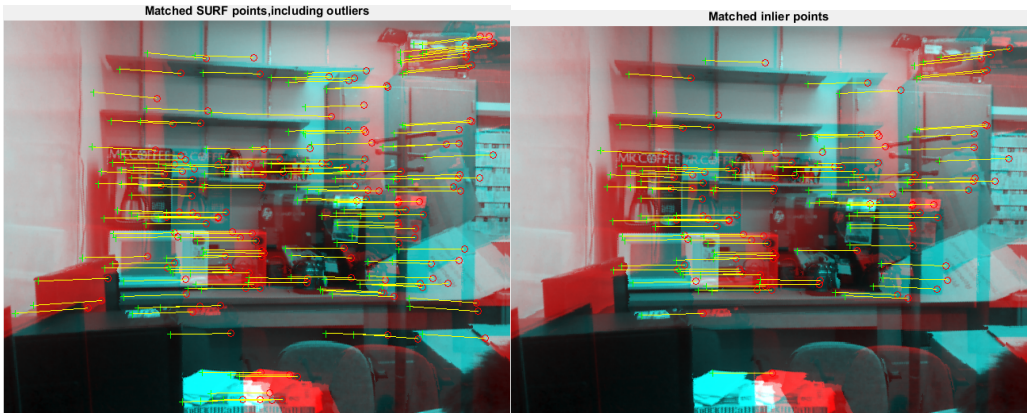


Figure 29: SURF feature matching with and without RANSAC

6.3 Implementation on iOS Platform

We realize the proposed indoor localization system on iOS platform and build an app to test the system performance. Thanks to Occipital Inc, they have developed a 3D structure sensor [81] which can be mounted on iPad (shown in Figure 30) to capture depth image and register it with color image captured by the original camera on the iPad, and generate 3D point cloud of the environment using their provided Application Program Interface (API). The test bed of our experiment includes a large corridor area and a small room area. Figure 31 shows a snapshot of our indoor localization app.

We compare pairwise error between the ground truth pose of the mobile device and its estimated pose in the 3D model.

We measure the error in 30 waypoints, the average error is around 10cm (translational) and 8 degree (rotational). However, due to the difficulties in implementation, the experiment is carried out by capturing online images from the iPad and post-processed in MATLAB.

Figure 30: Portable 3D sensor mounted on iPad

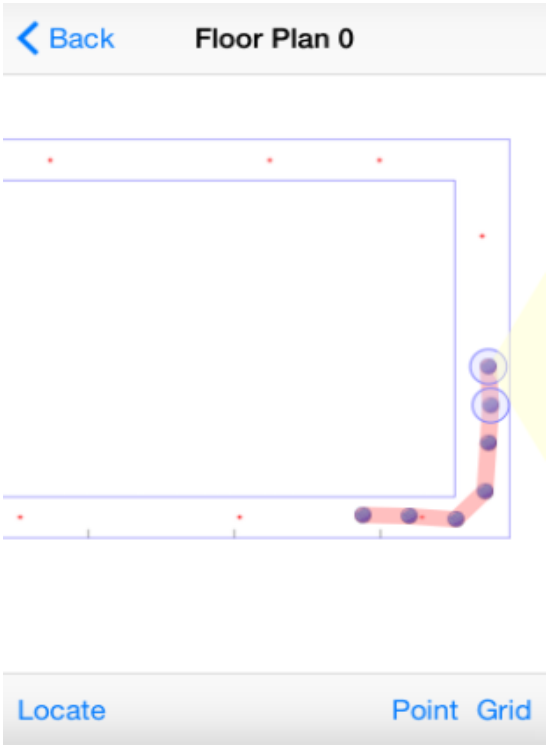


Figure 31: Snapshot of indoor localization app

6.4 Conclusion

In this chapter, we have presented an indoor pose localization system on mobile platform using portable 3D visual sensor. We first reduce the time consuming feature matching process by using WiFi RSS model to narrow down the search space from the whole 3D model to a small partition. Since we have a portable 3D sensor instead of just a 2D camera, we can bypass the 3D/2D or 2D/3D projection process and directly compare the online captured 3D point cloud with the local 3D model to find the pose estimation. As far as we know, this is the first attempt to use a portable 3D visual sensor to localize the mobile device in a 3D point cloud. The experiment carried out in the indoor environment encourages us to keep on working on this system.

In the future, we are looking to fully implement the system on mobile platform and test it in real time. And we will try to combine the system with more features like assistive technology to provide various location-based services to people with special needs.

CHAPTER SEVEN

PROBABILISTIC FRAMEWORK FOR MULTI-SENSOR FUSION BASED INDOOR LOCALIZATION ON MOBILE DEVICE

In Chapter Four, Five and Six, we have demonstrated the indoor localization methods based on WiFi RSS, motion sensors and visual sensors. Now we want to develop a probabilistic framework to fuse the information from different sensors.

7.1 Graph Structure Construction

The key idea of constructing a graph structure is to represent the indoor environment using a graph $G = (V, E)$, where V are vertices defined at each survey point during 3D modeling of the indoor environment, and E are edges that connect different segments of the 3D model if there is a direct access from one segment to another. The corresponded graph structure for a corridor area is shown in Figure 32 on the left. For a small room, we scan the entire room by standing in the middle and rotating

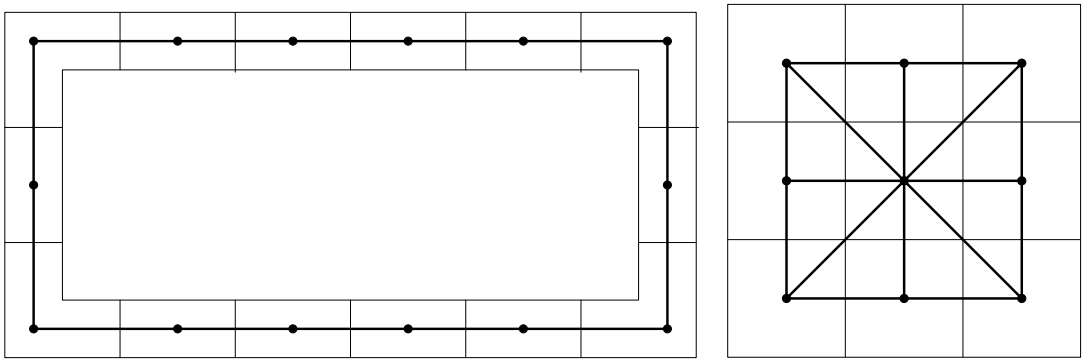


Figure 32: Graph structure construction.

the scanning system 360°. Thus, the room will be represented with a single vertex in the graph structure. For a large open space, the graph structure is shown in Figure 32 on the right. The black dot in the center of each small block corresponds to the survey point where we scan the indoor environment for visual features and WiFi RSS. Figure 33 illustrates the 3D structure on top of a 2D map.

A vertex V_i encodes the 3D color point cloud PT_{V_i}, WiFi received signal strength RSS_{V_i}, and location POS_{V_i}. These attributes are stored in an object array, $A_i = [PT_{V_i}, RSS_{V_i}, POS_{V_i}]$. An edge $E_{ij} = \{V_i, V_j\}$ connects two vertices, V_i and V_j. The edges act as constraints for the motion choices, since the user in vertex V_i can only directly access vertex V_j if there is an edge $E_{ij} = \{V_i, V_j\}$ between them.

By introducing a graph as the data structure, we are able to restrict the user movement and predict the user location at the next moment. Since the user can only move around connected vertices, we will only have limited amount of candidate vert

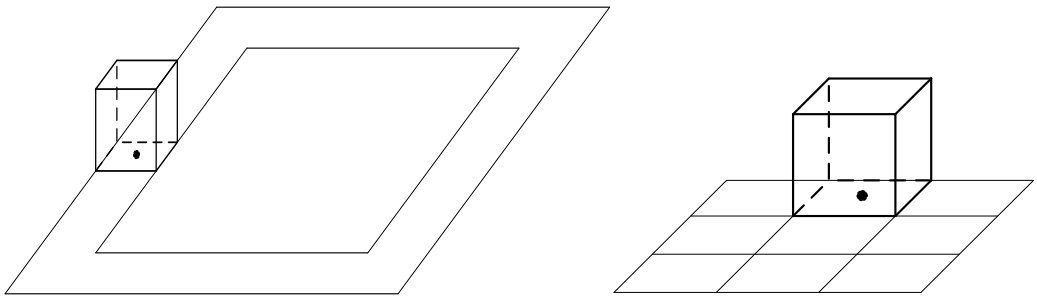


Figure 33: 3D structure on top of 2D map.

vertices for the next move. If the space involved in a vertex is small enough, finding the user's location is approximated as locating the vertex. This is similar to grid-based localization. However, the smaller the grid is, the higher the computational cost will be. In practice, we choose the grid size as the survey area covered during each 3D scan. Thus, locating the vertex gives a coarse estimation of the user's location. A finer localization within the vertex is achieved using particle filtering. The use of the graph structure also increases the system's robustness. In the case of crowded environments, where the sensor signal fidelity may not be reliable, the constraints in the graph can help detect a sensor failure if the prediction based on the sensor's measurement violates the constraints.

7.2 Hidden Markov Model

A general HMM characterizes a physical system with a state space model. In the problem of localization, the HMM model represents the temporal correlation of a user's location and orientation. Figure 34 shows a HMM factor graph, the state $X(t) = \{x(t), y(t), \theta(t), m(t), V(t)\}$, where $x(t), y(t), \theta(t)$ represents the user's location and orientation, $m(t) \in (walk, stop)$, $V(t)$ indicates the current vertex where the user is located in the graph structure. The state transition model $p(X_t \mid X_{t-1})$ constructed of $p(x_t, y_t \mid x_{t-1}, y_{t-1})$, $p(\theta_t \mid \theta_{t-1})$, $p(m_t \mid m_{t-1})$ and $p(V_t \mid V_{t-1})$ served as the motion model in our algorithm. The observation model is based on the WiFi signal strength measurement, motion sensor readings, and the captured image during online tracking.

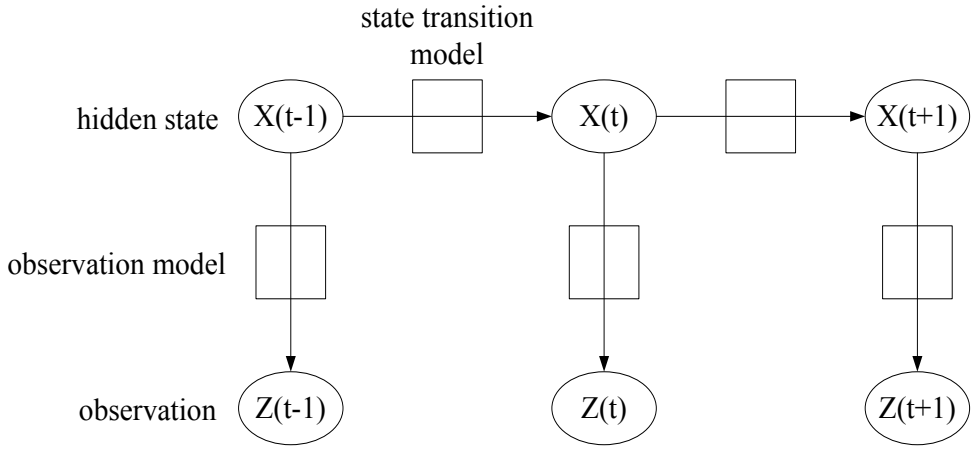


Figure 34: HMM model factor graph.

$p(m_t \mid m_{t-1})$ represents the probability of the motion state being walking or stopped given the previous motion state. It is defined as a 2 by 2 matrix, which models the preference of staying in the previous state, avoiding too-rapid changes in motion states. Moreover, we apply different matrices for different environments. This enables the system to model the fact that the user is more likely to stop in a room than in a corridor.

$p(\theta_t \mid \theta_{t-1})$ represents the probability of the current orientation θ_t given the previous orientation θ_{t-1}. It depends on whether the user is walking or stopped, and whether he is in a hallway or an open space. As the choice is limited to two, we use binary code to represent the situation. We define Walking = 1, Stopped = 0 for the leftmost binary digit, and Hallway = 1, Open space = 0 for the rightmost binary digit in the binary representation of the situation. In total we have four situations (00, 01, 10, 11). For example, if the user is walking in a hallway, his situation code is 11. If the next

motion state has been determined to be stop, the difference $\Delta\theta_t$ is sampled uniformly with a constant probability $\alpha \sim [0, 2\pi]$. Otherwise, if the next motion state is to be walking, we make a simple assumption that the user prefers to walk in a straight line, so that $\Delta\theta_t$ is sampled from a zero mean Gaussian distribution. If the user is in an open space, $\Delta\theta_t$ follows a unimodal Gaussian distribution. On the other hand, if the user is in a hallway, $\Delta\theta_t$ follows a bimodal Gaussian distribution, since the user has a higher chance of choosing from two opposite orientations.

$p(x_t, y_t \mid x_{t-1}, y_{t-1})$ represents the probability of the current location (x_t, y_t) given the previous location (x_{t-1}, y_{t-1}). If the previous motion state m_t has been determined as stopped, the current location is equal to the previous one. Otherwise, the current location is updated by sampling the moving distance d_t from a Gaussian $N(\mu, \sigma^2)$. Based on a simple straight motion assumption, the new location is calculated as follows:

$$x_t = x_{t-1} + d_t * \cos(\theta_t) \tag{23}$$

$$y_t = y_{t-1} + d_t * \sin(\theta_t) \tag{24}$$

$p(V_t \mid V_{t-1})$ represents the probability of the current vertex V_t given the previous vertex V_{t-1}. If V_t is in a corridor, we first calculate the walking/stopped probability. If $m_t = stopped$, then $V_t = V_{t-1}$. Otherwise, we calculate the distance that the user has travelled. For this distance, we determine whether the movement along the corridor results in a transition to another vertex, or remains within the same vertex area. The vertex transition is constrained to only two adjacent vertices. If V_t is in an open space, it

may have up to nine candidate vertices. We first sample the walking/stopped motion transition state and corresponding motion distance. Then, the vertex V_t is determined based on a simple straight-line movement.

The observation model describes the measurement likelihood of making an observation at different locations in the indoor environment. Our observation includes WiFi received signal strength, motion sensor readings, and captured images.

The WiFi signal strength measurement likelihood model uses the mean and variance of the signal at each location, calculated by Gaussian process regression.

$$p(z_t^{WiFi} \mid x_*) = \frac{1}{\sqrt{2\pi\sigma_{x_*}^2}} \exp(-\frac{(z_t^{WiFi} - \mu_{x_*})^2}{2\sigma_{x_*}^2}) \tag{25}$$

where z_t^{WiFi} is the received signal strength at time t, μ_{x_*} and $\sigma_{x_*}^2$ are the mean and variance at location x_*.

The camera measurement likelihood model is computed by pairwise pixel comparison between the current view and the view of a particle [11]. In the particle filter, a given particle's view of the environment can be projected from the 3D textured model

$$p(z_t^{Camera} \mid x_*) = \frac{\#\text{pixels in similar color}}{\#\text{total pixels}} \tag{26}$$

A predefined threshold determines the maximal color difference for two pixels, and a normalized color space is adopted to alleviate the effect of illumination.

The motion sensor measurement likelihood model compares the motion model $p(X_t \mid X_{t-1})$ with the motion dynamic model derived from motion sensor readings.

$$p(z_t^{Motion} \mid x_*) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{| z_t^{Motion} - x_* |^2}{2}) \quad (27)$$

where z_t^{Motion} is the pose calculated from the motion-sensor-based motion dynamic model, x_* is the pose estimated from the motion model defined in HMM.

7.3 Online Tracking with Particle Filter

Bayesian filtering is to estimate the posterior state X_t given all sensor measurements $Z_{0:t}$. Under the Markov assumption, we have the following recursive equation, which is updated whenever new sensor data becomes available:

$$p(X_t \mid Z_{0:t}) \propto p(Z_t \mid X_t) \int p(X_t \mid X_{t-1}) p(X_{t-1} \mid Z_{0:t-1}) dX_{t-1} \quad (28)$$

where $p(X_t \mid X_{t-1})$ represents the motion model, and $p(Z_t \mid X_t)$ is the observation model.

We implement Bayesian filtering using particle filter, which represents posterior over the state X_t by sets S_t of M weighted samples: $S_t = \{< X_t^{[m]}, w_t^{[m]} > | \, m = 1, ..., M\}$. Here each $X_t^{[m]}$ is a sample state, and $w_t^{[m]}$ is an importance weight of the state. Particle filter apply the recursive Bayesian filter update to estimate posteriors over the state space. Online tracking algorithm using particle filter is performed according to the following steps:

(1) Particle Initialization

The initial location is calculated through weighted K nearest neighbor (W-KNN) method. It searches for K closest matches of known locations in WiFi received signal space from the offline-built dataset. By averaging these K location candidates with adopting the distances in signal space as weights, the initial estimated location is acquired. This initial location estimation is used as the starting point for particles.

(2) Particle Propagation

Next, we apply the state transition model to guide the particle propagation. During the state transition process, we observe that the particle propagation plays an important role, which represents the state transition probability. The more accurate the particles propagate towards the right location, the better the localization performance will be. We generate new particles by sampling $X_t^{[m]}$ from the distribution $p(X_t \mid X_{t-1})$.

(3) Particle Weight Update

After the particle propagation in each epoch, we weight the sample $X_t^{[m]}$ by the probability $p(Z_t \mid X_t^{[m]})$. The weight is calculated as the product of different sensor measurement likelihood function. Then the weights $w_t^{[m]}$ of the samples are normalized so that they sum up to 1.

(4) Particle Resampling

Once the particle weights are updated, we perform importance resampling to update the particles' state by drawing with replacement a random sample $X_{t+1}^{[m]}$ from the sample set S_t according to the importance weight $w_t^{[m]}$. In resampling, the weight of each particle is treated as a probability where this particular particle is chosen to be the estimated location. Those particles with higher weights will be picked more frequently than others. This is how the resampling is able to eliminate those wrongly moved particles and correctly track the user location.

(4) Location Estimation

After the resampling process, the estimated location is calculated as the mean of all the resampled particles' location. To further increase the localization accuracy, we

perform Direct Linear Transformation (DLT) [83] between camera captured images in current location with projected image of the 3D model at the estimated location. The DLT parameters can be obtained using least square method. To solve for DLT parameters, we need at least 6 correspondences. The correspondences can easily be obtained using SIFT or SURF feature matching technique. The matching process frequently contains "outliers", therefore, we apply the well-known RANSAC algorithm to filter out the "outliers". After getting the DLT parameters, the camera location can be calculated by solving the projection matrix. This process is able to refine the location estimation, but due to its high computational cost, we only invoke it in certain key vertices predefined in the graph structure, for example, at the corridor corner. The user can also invoke the process at their wish when they require a better localization performance. The process will also wake up after a time period T to correct the estimation error. Table 6 gives a pseudo-code of the particle filter algorithm.

7.4 Implementation on iOS Platform & Experimental Analysis

We realize the indoor localization algorithm on the iOS platform and build an app to test the system performance. The system workflow is shown in Figure 35. During the offline training phase, WiFi received signal strength, color images, point cloud, and motion signals, including user acceleration, device attitude, and rotation rate, are recorded along the entire scenario. A detailed 3D model of the indoor environment is generated by fusing color images from the camera and point cloud from LiDAR. Then,

Table 6

Pseudo-code of particle filter.

$$\left\{X_t^{[m]}, w_t^{[m]}\right\}_{m=1}^{M} = PF\left[\left\{X_{t-1}^{[m]}, w_{t-1}^{[m]}\right\}_{m=1}^{M}, Z_t\right]$$

Initialization $\left[\left\{X_{t=0}^{[m]}, w_{t=0}^{[m]}\right\}_{m=1}^{M}\right]$

FOR $m = 1 : M$

Particle propagation $X_t^{[m]} \sim p\left(X_t \left| X_{t-1}^{[m]}\right.\right)$

Update weight using observation $w_t^{[m]} = p\left(Z_t \left| X_t^{[m]}\right.\right)$

ENDFOR

Normalize weights to $\sum_{m=1}^{M} w_t^{[m]} = 1$

$\left\{X_t^{[m]}, w_t^{[m]}\right\}_{m=1}^{M} = \text{Resample}\left[\left\{X_t^{[m]}, w_t^{[m]}\right\}_{m=1}^{M}\right]$

we apply Gaussian process modeling to generate a signal strength map of the WiFi signal strength map is overlapped with the 3D model and we divide the 3D model into different segments according to our survey points. Each segment is encoded in the vertices of a graph structure. We store the graph structure in the mobile device for the online tracking phase. When the “Locate” button is pressed, the mobile device starts scanning the WiFi received signal strength from all the access points it can detect. Particles are initialized through weighted the K nearest neighbor method. After the initial distribution, particles start to propagate under the guidance of the HMM motion model. Every 5 seconds, the

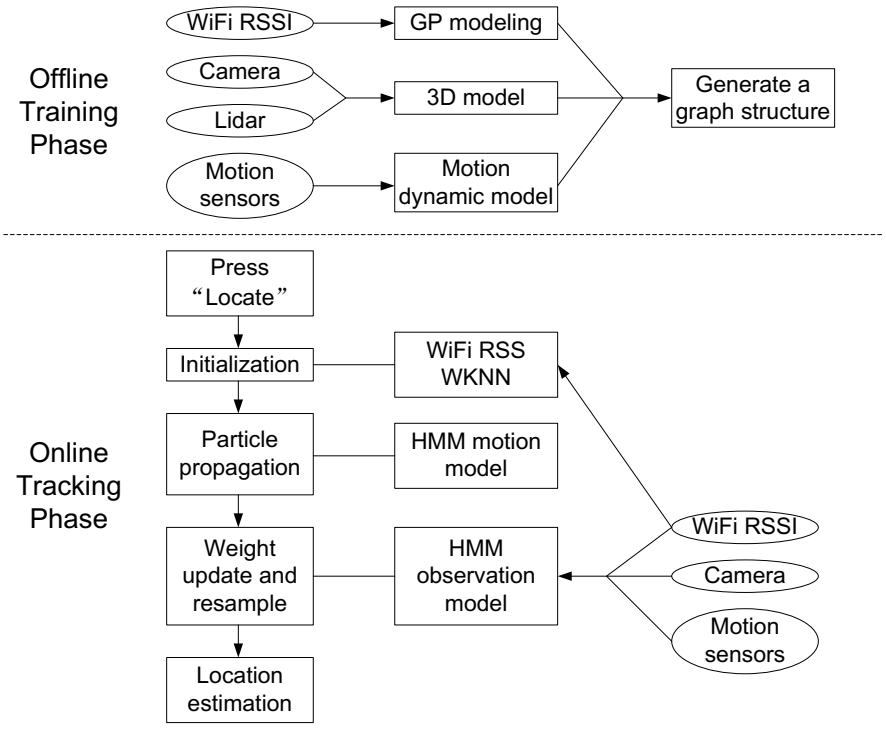


Figure 35: System workflow.

particles are resampled using the HMM observation model. Through particle filtering, we are able to locate the user and track the user's movement in real time.

The training and testing are conducted in an office building corridor area and a library's open space. In total, we have 24 survey points in the corridor and 21 survey points in the library. Localization tests are conducted on a predefined path. The average length of the path is about 90 m in the library's open space and 100 m in the corridor area. The location update is performed every 5 s, which means the particle filtering step can be completed in 5 s. The memory usage is under 40 MB RAM since the iOS app will be forced to shut down if it exceeds this limit. During the traverse on the path, we

measured the error distance between the estimated location and the ground truth location. The ground truth location is based on manual annotation of waypoints. Whenever the tester reaches a waypoint, the timestamp is recorded and the estimated location with the actual one are compared. Figure 36 shows screenshots of the real time test results in a corridor area.

Figure 37 illustrates the localization error in each waypoint. By including the motion sensors, the error has dropped significantly compared to using only the WiFi based localization method.

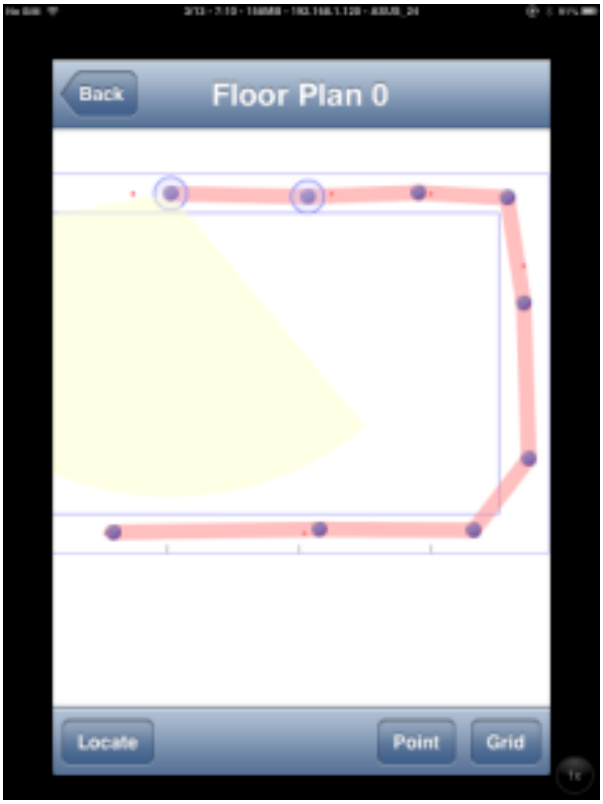


Figure 36: Screen shots of localization test.

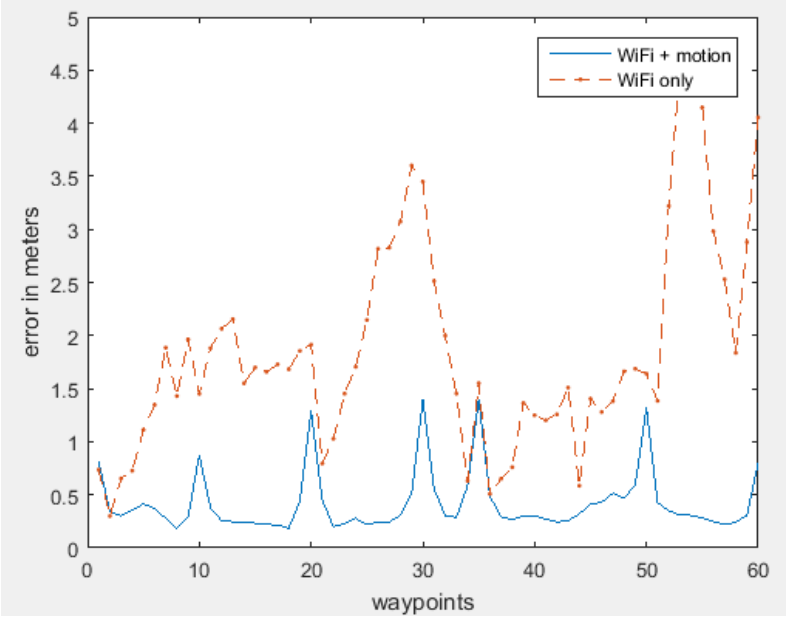


Figure 37: Error in each waypoint.

To gain insight into the localization error distribution, Figure 38 presents the cumulative probabilities of the localization errors of the different cases. The comparison further shows that the motion dynamic model greatly increases localization accuracy.

Due to the implementation issue on the iOS device, we have not included the visual sensors in the real time test. The visual sensors are applied separately. After the real time test is done, we capture the images at all the waypoints and correct the estimated location. Table 8 shows the correction results.

Table 7 shows that the visual sensors could further improve the localization accuracy. Overall, the localization performance has proved that our iOS application is a robust, accurate, highly integrated indoor localization system. However, we have not implemented various state-of-the-art indoor localization methods in the literature, due to

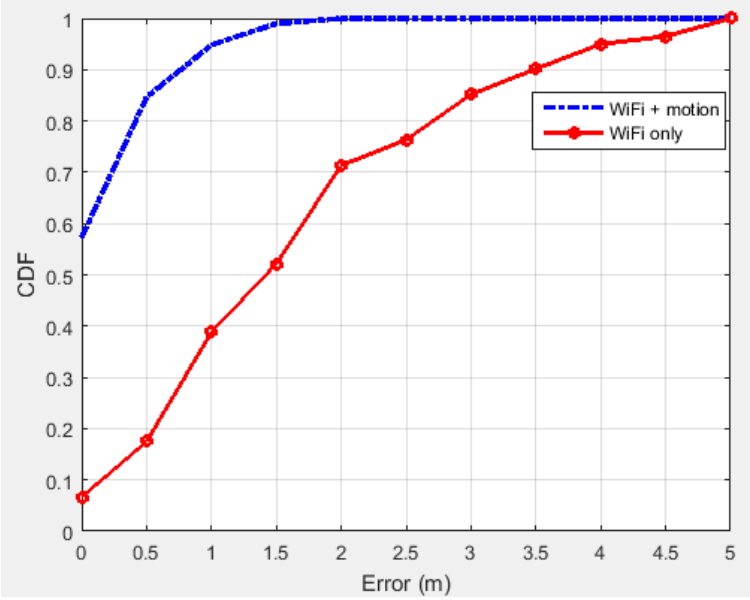


Figure 38: Cumulative distribution function (CDF) of error.

Table 7

Visual sensor correction on localization results.

Error	Mean	Median	Maximum
Visual sensor correction	0.10 m	0.23 m	0.66 m

the difficulty in realizing them on a mobile platform. The state-of-the-art method of a WiFi signal strength based method using Gaussian process is discussed in Chapter Four, we implemented this on the iOS platform. Moreover, we introduce a framework based on HMM to effectively fuse the WiFi information with motion sensor and visual sensor information on the mobile platform in order to improve system performance.

7.7 Conclusion

In this chapter, we have demonstrated an indoor localization system based on a graph structure and multimodal particle filtering technique. The implementation on the iOS platform, and the test in a real world situation proved that our application is a reliable indoor localization system. To the best of our knowledge, this is the first iOS app delivering such accurate, highly integrated indoor localization system on a small mobile device. Based on our system, many location-aware applications will be able to function properly indoors, providing more convenient service to people's daily lives.

In the future, we will focus on fusing the visual sensors on board into our real time localization system. We believe the key technology for future localization systems lies in how to efficiently fuse the information provided by various sensors. We are also looking to integrate our system into other platforms, for example, a vehicular infotainment platform, so that we have access to more sensor information and can function both indoors and outdoors.

CHAPTER EIGHT

SUMMARY & FUTURE WORK

Part of this dissertation work concentrates on indoor localization on smart mobile device. In our work, we explore the approaches of using WiFi RSS, motion sensors and visual sensor to achieve the goal of localizing the mobile device user and tracking the user movement at the same time. Through simulation in MATLAB and implementation on iOS platform, we analyze the performances (accuracy, robustness and time complexity) of different approaches. Furthermore, we derive a probabilistic framework using graph theory, HMM and particle filter to fuse the information from multiple sensors, so that we can easily combine additional sensors on board to help improve the localization system performance on smart mobile device.

A good direction for future research is to investigate the deep neural network architecture for indoor localization. Deep learning has proved its ability in various tasks like image classification and object detection within videos. The multi-layer neural network is capable of extracting meaningful features from the training samples to fulfill the assigned task. Alex et al from University of Cambridge have developed a deep convolutional neural network called PoseNet for 6–DoF camera pose localization [84]. The basis of PoseNet is a famous deep neural network architecture for classification, the GoogLeNet [85]. GoogLeNet is a 22-layer convolutional network. PoseNet adds one extra layer to enable the ability of pose regression. It operates in real time, and obtains approximately 2 meters and 3 degrees accuracy for large outdoor scenes.

Additionally, one can sidestep the need for millions of training images by use of transfer learning, which is a very powerful tool in deep learning. The transfer learning is based on a pre-trained model that trained on the ImageNet database with tens of millions of images. People can take this pre-trained convolutional neural network (CNN), remove the last fully-connected layer, and treat the rest as a fixed feature extractor for a new dataset. Researchers have shown that the features learned from deep CNN on a very large database like ImageNet outperformed most of the famous expert-crafted features, e.g. SIFT and SURF [86]. People can also fine-tune the weights of the pre-trained network. This allows researchers to use a much smaller dataset and achieve state-of-the-art image recognition results.

REFERENCES

[1] G. Fischer, B. Dietrich, F. Winkler, "Bluetooth indoor localization system" in *Proceedings of the 1st Workshop on Positioning, Navigation and Communication*, Hannover, Germany, March 2004, pp. 147-156.

[2] G. Jin, X. Lu, M. S. Park, "An indoor localization mechanism using active RFID tag" in *Proceedings of International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing*, Taichung, Taiwan, June 2006, pp. 40-43.

[3] M. Sugano, T. Kawazoe, Y. Ohta, M. Murata, "Indoor localization system using RSSI measurement of wireless sensor network based on ZigBee standard" in *Proceedings of Wireless and Optical Communication Multi Conference*, Alberta, Canada, July 2006, pp. 1-6.

[4] B. Ferris, D. Hahnel, D. Fox, "Gaussian processes for signal strength-based location estimation" in *Proceedings of Robotics: Science and Systems,* Philadelphia, PA, USA, August 2006, pp. 16-19.

[5] F. Duvallet, A. Tews, "WiFi position estimation in industrial environments using Gaussian processes" in *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, October 2009, pp. 2216-2221.

[6] J. Machaj, R. Piche, P. Brida, "Rank based fingerprinting algorithm for indoor positioning" in *Proceedings of International Conference on Indoor Positioning and Indoor Navigation*, Guimaraes, Portugal, September 2011, pp. 6-11.

[7] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors" in *Proceedings of ACM Conference on Ubiquitous Computing*, Pittsburgh, PA, USA, September 2012, pp. 5-8.

[8] J. Liu, R. Chen, L. Pei, R. Guinness, H. Kuusniemi, "A hybrid smartphone indoor positioning solution for mobile LBS" in *Sensors*, December, 2012, vol. 12, pp. 17208–17233.

[9] M. Werner, M. Kessel, C. Marouane, "Indoor positioning using smartphone camera" in *Proceedings of International Conference on Indoor Positioning and Indoor Navigation*, Guimaraes, Portugal, September 2011, pp. 1-6.

[10] I. Arai, S. Horimi, N. Nishio, "Wi-Foto 2: Heterogeneous device controller using WiFi positioning and template matching" in *Proceedings of Pervasive*, Helsinki, Finland, May 2010.

[11] Y. Fu, S. Tully, G. Kantor, H. Choset, "Monte Carlo localization using 3D texture maps" in *Proceedings of International Conference on Intelligent Robots and Systems*, San Francisco, USA, September 2011, pp. 482-487.

[12] J. Liang, N. Corso, E. Turner, A. Zakhor, "Image based localization in indoor environments" in *Proceedings of Computing for Geospatial Research and Application*, San Jose, USA, July 2013, pp. 70-75.

[13] J. Wolf, W. Burgard, H. Burkhardt, "Robust vision-based localization by combining an image-retrieval system with Monte Carlo localization" in *IEEE Trans. Robot*, 2005, vol. 21, pp. 208–216.

[14] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints” in *Int. J. Comput. Vis,* 2004, vol. 60, pp. 91–110.

[15] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, “Speeded-Up Robust Features (SURF)” in *J. Comput. Vis. Image Underst,* 2008, vol. 110, pp. 346–359.

[16] F. Hoflinger, R. Zhang, J. Hoppe, A. Bannoura, L. Reindl, J. Wendeberg, M. Buhrer, C. Schindelhauer, “Acoustic self-calibrating system for indoor smartphone tracking (assist)” in *Proceedings of International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Sydney, Australia, 2012, pp. 1–9.

[17] E. Drakopoulos, C. Lee, “Optimum multisensor fusion of correlated local decisions” in *IEEE Trans. Aerosp. Electron. Syst*, 1991, vol. 27, pp. 593–606.

[18] D. Ciuonzo, G. Romano, P. S. Rossi, “Optimality of received energy in decision fusion over Rayleigh fading diversity MAC with non-identical sensors” in *IEEE Trans. Signal Process.* 2013, vol. 61, pp. 22–27.

[19] A. Ruiz-Ruiz, P. Lopez-de-Teruel, O. Canovas, “A multisensory LBS using SIFT-based 3D models” in *Proceedings of International Conference on Indoor Positioning and Indoor Navigation*, Sydney, Australia, November 2012, pp. 13-15.

[20] J. Kim, Y. Kim, S. Kim, “An accurate localization for mobile robot using extended Kalman filter and sensor fusion” in *Proceedings of IEEE International Joint Conference on Neural Networks*, Hong Kong, China, June 2008, pp. 2928-2933.

[21] R. Zhang, A. Bannoura, F. Hoflinger, L. M. Reindl, C. Schindelhauer, "Indoor localization using a smart phone" in *Proceedings of the IEEE Sensors Applications Symposium (SAS)*, Galveston, TX, USA, February 2013, pp. 19–21.

[22] S. Beauregard, Widyawan, M. Klepal, "Indoor PDR performance enhancement using minimal map information and particle filters" in *Proceedings of Position, Location and Navigation Symposium*, Monterey, CA, USA, May 2008, pp. 5-8.

[23] M. Quigley, D. Stavens, A. Coates, S. Thrun, "Sub-meter indoor localization in unmodified environments with inexpensive sensors" in *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 2010, pp. 2039-2046.

[24] B. Limketkai, D. Fox, L. Liao, "CRF-Filters: Discriminative particle filters for sequential state estimation" in *Proceedings of Robotics and Automation*, Roma, Italy, 10 April 2007.

[25] Z. Xiao, H. Wen, A. Markham, N. Trigoni, "Lightweight map matching for indoor localization using conditional random fields" in *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, Berlin, Germany, April 2014.

[26] C. F. Xu, W. D. Geng, Y. H. Pan, "Review of Dempster-Shafer method for data fusion" in *Chinese Journal of Electronics.* 2001, vol. 29, pp. 393–396.

[27] P. Kasebzadeh, G. S. Granados, E. S. Lohan, "Indoor localization via WLAN path-loss models and Dempster-Shafer combining" in *Proceedings of Localization and GNSS*, Helsinki, Finland, June 2014.

[28] X. He, S. Badiei, D. N. Aloi, J. Li, "WiFi iLocate: WiFi based indoor localization for smartphone" in *Proceedings of Wireless Telecommunication Symposium*, Washington, DC, USA, August 2014, pp. 1-7.

[29] X. He, D. N. Aloi, J. Li, "WiFi based indoor localization with adaptive motion model using smartphone motion sensors" in *Proceedings of the International Conference on Connected Vehicle and Expo (ICCVE)*, Vienna, Austria, November 2014, pp. 786-791.

[30] X. He, D. N. Aloi, J. Li, "Portable 3D visual sensor based indoor localization on mobile device" in *Proceedings of the Consumer Communication & Networking Conference (CCNC)*, Las Vegas, USA, Janaury 2016, pp. 1125-1128.

[31] X. He, D. N. Aloi, J. Li, "Probabilistic multi-sensor fusion based indoor positioning system on a mobile device" in *Sensors*, December 2015, vol. 15, pp. 31464-31481.

[32] M. Huang, S. Dey, "Dynamic quantizer design for hidden Markov state estimation via multiple sensors with fusion center feedback" in *IEEE Trans. Signal Process.* 2006, vol. 54, pp. 2887–2896.

[33] P. Salvo Rossi, D. Ciuonzo, T. Ekman, "HMM-based decision fusion in wireless sensor networks with noncoherent multiple access" in *IEEE Commun. Lett.* 2015, vol. 19, pp. 871–874.

[34] H. Blom, Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markov switching coefficients" in *IEEE Trans. Autom. Control* 1988, vol. 33, pp. 780–783.

[35] http://spreo.co/.

[36] S. Chumkamon, P. Tuvaphanthaphiphat, P. Keeratiwintakorn, “A blind navigation system using RFID for indoor environments” in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, 2008, Thailand, pp. 765-768.

[37] E. Chan, G. Baciu, *Introduction to Wireless Localization*, John Wiley & Sons Singapore Pte. Ltd 2012.

[38] S. Godha, G. Lachapelle. M. E. Cannon, “Integrated GPS/INS system for pedestrian navigation in a signal degraded environment” in *Proceedings of the 19th International Technical Meetings Satellite Division Institute Navigation*, Fort Worth, TX, USA, September 2006, pp. 2151–2164.

[39] I. Skog, P. Handel, J. Nilsson, J. Rantakokko, “Zero-velocity detection—An algorithm evaluation” in *IEEE Trans. Biomed. Eng*, 2010, vol. 57, pp. 2657–2666.

[40] S. Y. Cho, C. G. Park, “MEMS based pedestrian navigation system” in *J. Navig*. 2006, vol. 59, pp. 135–153.

[41] C. Ascher, C. Kessler, M. Wankerl, G. F. Trommer, “Dual IMU indoor navigation with particle filter based map-matching on a smartphone” in *Proceedings of the IEEE International Conference Indoor Positioning Indoor Navigation (IPIN)*, Zurich, Switzerland. September 2010, pp. 1–5.

[42] J. C. Alvarez, D. Alvarez, A. López, R. C. González, “Pedestrian navigation based on a waist-worn inertial sensor”, in *Sensors*. 2012, vol. 12, pp. 10536–10549.

[43] K. C. Lan, W. Y. Shih, “Using simple harmonic motion to estimate walking distance for waist-mounted PDR” in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Paris, France, April 2012, pp. 2445–2450.

[44] A. Mikov, A. Moschevikin, A. Fedorov, A. Sikora “A localization system using inertial measurement units from wireless commercial handheld devices” in *Proceedings of the IEEE International Conference Indoor Positioning Indoor Navigation (IPIN)*, Montbeliard, France, October 2013, pp. 1–7.

[45] J. Liu, R. Chen, L. Pei, R. Guinness, H. Kuusniemi, “A hybrid smartphone indoor positioning solution for mobile LBS” in *Sensors*. 2012, pp. 17208–17233.

[46] N. Kothari, B. Kannan, M. B. Dia, “Robust Indoor Localization on a Commercial Smart-Phone” in Carnegie-Mellon University, Pittsburgh, PA, USA, 2011, Technical Report CMU-RI-TR-11–27.

[47] N. Ravi, P. Shankar, A. Frankel, A. Elgammal, L. Iftode, “Indoor localization using camera phones” in *Mobile Computing Systems and Applications*, 2006.

[48] J. Liang, N. Corso, E. Turner, A. Zakhor, “Image based localization in indoor environments” in *Computing for Geospatial Research and Application (COM. Geo)*, 2013, pp. 70-75.

[49] A. Ruiz-Ruiz, P. Lopez-de-Teruel, O. Canovas, “A multisensory LBS using SIFT-based 3D models” in *Proceedings of International Conference on Indoor Positioning and Indoor Navigation*, Sydney, Australia, November 2012, pp. 1-10.

[50] Y. Lee, G. Medioni, “Wearable rgbd indoor navigation system for the blind” in *Workshop on Assistive Computer Vision and Robotics, ECCV*, 2014, pp. 493-508.

[51] Y. Bar-Shalon, “Multi-target multi-sensor tracking” in Artec House, Norwood, 1990.

[52] W. Sorensen, “Special issue on the applications of the Kalman filter” in *IEEE Trans. Autom. Control*, 1983.

[53] G, Welch, G. Bishop, “An Introduction to the Kalman Filter” in Department of Computer Science, University of North Carolina, Tech. Rep. TR-95-041, 1995.

[54] S. J. Julier, J. K. Uhlmann, “A new extension of the Kalman filter to nonlinear systems” in *International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, 1997, pp. 182–193.

[55] L.D. Stone, T.L. Corwin, C.A. Barlow, “Bayesian Multiple Target Tracking” in Artech House Inc, Norwood, MA, 1999.

[56] N.J. Gordon, D.J. Salmond, A.F.M. Smith, “Novel approach to nonlinear/non Gaussian Bayesian state estimation” in *IEEE Proceedings – F*, 1993, pp. 107–113.

[57] G. Shafer, “A Mathematical Theory of Evidence” in Princeton University Press, 1977.

[58] T.D. Garvey, J.D. Lowrance, M.A. Fischler, “An inference technique for integrating knowledge from disparate sources” in *Proc. of the International Joint Conference on Artificial Intelligence*, 1981, pp. 319–325.

[59] J.A. Barnett, “Computational methods for a mathematical theory of evidence” in *Proc. of the International Joint Conference on Artificial Intelligence*, 1981, pp. 868–875.

[60] P. Bahl, V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system" in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM' 00)*, March 2000, pp. 775-784.

[61] M. Youssef, "HORUS: A WLAN-based indoor location determination system" in Department of Computer Science, University of Maryland, 2004.

[62] T. King, S. Kopf, T. Haenselmann, C. Lubberger, W. Effelsberg, "Compass: A probabilistic indoor positioning system based on 802.11 and digital compasses" in *Proceedings of the 1st international workshop on Wireless network test beds, experimental evaluation & characterization*, *ACM*, 2006, pp. 34-40.

[63] http://www.shopkick.com/.

[64] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen and F. Ye, "Push the Limit of WiFi based Localization for Smartphones" in *ACM MobiCom* 2012.

[65] L. Agrawal, D. Toshniwal, "Smart Phone Based Indoor Pedestrian Localization System" in *13th International Conference on Computational Science and Its Application*, Ho Chi Minh City, Vietnam, 2013, pp. 137-143.

[66] A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann, "GPPS: A Gaussian Process Positioning System for Cellular Networks" in *Advances in Neural Information Processing Systems*, 2003.

[67] C. E. Rasmussen and C. K. I. Williams, "Gaussian Processes for Machine Learning" in MIT Press, 2006.

[68] B. Ferris, D. Fox and N. Lawrence, "WiFi-SLAM using Gaussian process latent variable models" in *Proceedings of IJCAI 2007*, Hyderabad, India, 2007, pp. 2480-485.

[69] H. Leppakoski, J. Collin, and J. Takala., "Pedestrian navigation based on inertial sensors, indoor map, and WLAN signals" in *Proceedings of Acoustics, Speech and Signal Processing*, Kyoto, Japan, March 2012, pp. 1569-1572.

[70] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, F. Zhao., "A reliable and accurate indoor localization mthod using phone inertial sensors", in *Proceedings of International Conference on Ubiquitous Computing*, Pittsburgh, USA, September 2012.

[71] F. Evennou, F. Marx, "Advanced integration of WiFi and inertial navigation systems for indoor mobile positioning", in *EURASIP Journal on Applied Signal Processing*, January 2006, pp. 1-11.

[72] M. Weyn, M. Klepal and Widyawan., "Adaptive motion model for a smartphone based opportunistic localization system" in *Proceedings of Mobile Entity Localization and Tracking in GPS-less environments*, Orlando, USA, September 2009, pp. 50-65.

[73] A. Rai, K. Chintalapudi, V. Padmanabhan and R. Sen., "Zee: Zero-Effort Crowdsourcing for Indoor Localization" in *Proceedings of the 18th annual international conference on Mobile computing and networking*, Istanbul, Turkey, August 22-26, 2012.

[74] Jeong Won Kim., "A step, stride and heading determination for the pedestrian navigation system" in *Journal of global positioning system*, pp. 273-279.

[75] https://www.google.com/atap/projecttango/.

[76] C. Jaramillo, I. Dryanovski, R. Valenti, J. Xiao, “6-DoF Pose Localization in 3D Point-Cloud Dense Maps Using a Monocular Camera” in *Proceedings of IEEE International Conference on Robotics and Biomimetics*, Shenzhen, China, December 2013, pp. 1747-1752.

[77] P. J. Besl, N.D. McKay, "A Method for Registration of 3-D Shapes" in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1990, pp. 239-254.

[78] Martin A. Fischler, Robert C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography” in *Proc. Image Understanding Workshop*, April 1980, pp. 71-88.

[79] H. Alismail, L. D. Baker, B. Browning, “Automatic calibration of a range sensor and camera system” in *Proceedings of Second Joint 3DIM/3DPVT Conference*, Zurich, Switzerland, October 2012, pp. 286-292.

[80] V. Lepetit, F. Moreno-Noguer and P. Fua, “EPnP: An Accurate O(n) Solution to the PnP Problem” in *International Journal Of Computer Vision*, 2009, pp. 155-166.

[81] http://www.structure.io/.

[82] F. Endres, J. Hess, D. Cremers, and N. Engelhard, “An Evaluation of the RGB-D SLAM System,” in *Proceedings of International Conference on Robotics and Automation*, St Paul, USA, May 2012, pp. 1691–1696.

[83] http://kwon3d.com/theory/dlt/dlt.html.

[84] A. Kendall, M. Grimes, R. Cipolla, “PoseNet: a convolutional network for real-time 6-dof camera relocalization” in *Proceedings of the International Conference on Computer Vision (ICCV)*, Santiago, Chile, December 2015.

[85] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, "Going deeper with convolutions" in *Proceedings of the Computer Vision and Pattern Recognition (CVPR) conference*, Boston, USA, June 2015.

[86] A. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition" in *Proceedings of the Computer Vision and Pattern Recognition (CVPR) conference*, Washington DC, USA, June 2014, pp. 512-519.