Sensor-Data Fusion for Multi-Person Indoor Location Estimation

by

Parisa Mohebbi

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

Master of Science

Department of Computing Science

University of Alberta

© Parisa Mohebbi, 2018

Abstract

We consider the problem of estimating the location of people as they move and work in indoor environments. More specifically, we focus on the scenario where one of the persons of interest is unable or unwilling to carry a smartphone, or any other "wearable" device, which frequently arises in caregiver/cared-for situations. We consider the case of indoor spaces populated with anonymous binary sensors (Passive Infrared motion sensors) and eponymous wearable sensors (smartphones interacting with Estimote beacons), and we propose a solution to the resulting sensor-fusion problem. Using a data set with sensor readings collected from one-person and two-person sessions engaged in a variety of activities of daily living, we investigate the relative merits of relying solely on anonymous sensors, solely on eponymous sensors, or on their combination. We examine how the lack of synchronization across different sensing sources impacts the quality of location estimates, and discuss how it could be mitigated without resorting to device-level mechanisms. Finally, we examine the trade-off between the sensors' coverage of the monitored space and the quality of the location estimates.

Preface

The data gathering part of this project, which is described in chapter 3.2.6, received research ethics approval from the University of Alberta Research Ethics Board, Project Name "Recognition of Daily Activities Using Environment-based Sensing: Simulating and Recognizing Activities in the Smart Condo", Study ID Pro00073382, 6/12/2017.

Parts of the chapters 1, 2, section 3.2, chapters 3.2.6, and 5 has been published as Mohebbi, P.; Stroulia, E.; Nikolaidis, I. "Sensor-Data Fusion for Multi-Person Indoor Location Estimation". Sensors 2017, 17, 2377.

Section 3.1 of this thesis has been published as Mohebbi, P.; Stroulia, E.; Nikolaidis, I. "Indoor Localization: A Cost-Effectiveness vs. Accuracy Study". Computer-Based Medical Systems (CBMS), 2017 IEEE 30th International Symposium on, IEEE, 2017, Page(s):552 - 557.

In both these publications, my contribution was implementing the method, performing experiments, analyzing the results and preparing the paper draft. Prof. Eleni Stroulia and Prof. Ioanis Nikolaidis, my supervisors, advised me through both works with ideas on how to improve the methods and also preparing the final papers. In the first publication, Aristotelis Kolias and Colton Begert has done the installing and setting up the data collection processes for PIR motion sensors.

To Sina
For standing beside me and encouraging me
through all my failures and successes.

And to my parents and my brother
For their endless love and support.

Acknowledgements

I would like to express my utmost gratitude to both my supervisors, Prof. Eleni Stroulia and Prof. Ioanis Nikolaidis for their valuable guideline and incredible patience. Their constructive feedbacks and invaluable supervision directed me through this project. I can not imagine this project done without their priceless assistance and excellent knowledge.

I also like to thank Lili Liu and Christine Daum, AGE-WELL WorkPackage6 (WP6) co-lead and coordinator correspondingly, who were instrumental in developing the data-collection and the ethics protocols under which the data set used for this study was collected.

I would also like to thank Mr. Aristotelis Kolias and Mr. Colton Begert who helped a lot in data gathering session by setting up the motion sensor; and preparing the final version of the $SmartCondo^{TM}$software. I also thank Mr. Spencer Plant who developed the Android application implemented for holding the activity protocol we gave our participants for the data gathering sessions described in section 3.2. I would also like to thank Mr. Seann Murdock for developing the iOS application used for experiments in section 3.1

This work was funded by the AGE-WELL NCE, under the WorkPackage6 activity, and the NSERC Discovery Grant program.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

According to a 2012 study commisioned by the Alzheimer's Society of Canada, 747,000 Canadians have some type of cognitive impairment, including dementia, and this number is expected to double by 2031. People with dementia experience challenges with daily activities (e.g., cooking meals, ironing, taking medication, personal care), such as misplacing materials and failing to complete tasks in the right sequence. Having accurate information about an older adult's daily activities, and the patterns of these activities, can provide rich information on his/her abilities and capacity for functional independence. Major deviations in daily patterns should likely be considered as indicators of a person's physical, cognitive and/or mental decline. Having such information could alert caregivers of potentially risky events and the need for additional support.

The advancing wave of Internet-of-Things technologies holds immense promise for enabling such data collection and analysis and for delivering appropriate support. In the $SmartCondo^{TM}$ project, we have been developing a sensor-based platform for non-intrusively monitoring people at home, analyzing the collected data to extract information about the occupants' activities, simulating the extracted information in a 3D virtual world, and generating recommendations for themselves and their caregivers. To meet the non-obtrusiveness requirement of our platform, we have excluded from $SmartCondo^{TM}$ any image and video capture devices. Of course, for the sake of reconstructing the ground truth via manual annotation, the experiments we carried out also in-

cluded video cameras. However, in a production-environment deployment, no cameras would be used.

This hardware–software platform has been installed in the *SmartCondo*^{TM} simulation space, a dedicated teaching-and-research space in the University of Alberta's Edmonton Clinic Health Academy (ECHA). The *SmartCondo*^{TM}is a fully functional apartment with one bedroom, a bathroom, and an open kitchen-and-living space. Infused into the apartment and its furnishings are sensors that record a variety of environmental variables (i.e., levels of light and sound, temperature, and humidity) as well as the activities of the occupant(s) (i.e., their motion and use of furniture, cabinetry, and appliances). The data acquired by these sensors is transmitted and analyzed into a central cloud-based repository. The *SmartCondo*^{TM}has recently been redesigned to include Bluetooth Low-Energy (BLE) beacons attached to different objects in the apartment. The occupants can be provided with a smartphone, running a background service that collects, and transmits to the *SmartCondo*^{TM}platform, signal-strength measurements from any nearby BLE beacons. These two types of data sources—sensors and beacons—are used to infer the occupants' locations at each point in time. The server generates textual reports, spatial visualizations, and 3D virtual-world simulations for the inferred movements and activities of every occupant. In addition, it can potentially generate alerts for special incidents, which can be sent to the occupants' caregivers, or anyone of their choice.

In the previous works that have been done on the *SmartCondo*^{TM}project in the software engineering and network labs, the trade-offs between the accuracy of the location-estimation process for one occupant, based on PIR (Pyroelectric or "Passive" Infrared) sensors only vs. the overall cost of the sensor installation has been investigated [36]. Next, a study has been done on how the use of RFIDs in addition to PIRs could be exploited to enable location recognition for multiple occupants [4]. In this project, we investigated the use of BLE sensors (Estimote stickers and beacons[1]) for the purpose of indoor localization and activity recognition. Moving one step further, we studied the possibility

[1]https://www.estimote.com/

of fusing Estimote stickers data with PIR motion sensors to do multi-person localization.

Multi-person location estimation, as the first step to activity recognition, is a challenging problem and has received relatively little attention in the literature. This is partly due to the implicit assumption that if the subjects carry with them active (hence, eponymous) devices, each person can be localized independently, in isolation of the rest; hence, any method for estimating the location of a single individual is assumed to generalize to multiple individuals. However, the situation is drastically different when one (or more) of the subjects do not systematically carry/wear such a device, either because they cannot, they do not want to, or they simply forget to – typical of many care-delivery scenarios. Estimating the location of an individual does not yield the same results when applied to a scenario when the individual is alone vs. when the individual is one among many in the observed space. For example, the radio frequency (RF) propagation environment in a given space varies over time because of the dynamics of human and object placement within that space. In fact, [38] has utilized the impact of humans on the RF environment to estimate the locations of multiple subjects, based on models of how the fingerprints of radio signal strength indicators (RSSIs) change in the presence of people in the space. Nevertheless, this method requires a large set of transmitters and receivers to cover the entire area in each room and the placement of the transmitters/receivers needs to be exact to ensure that they are in the line of sight (LoS). We address more of the related work in the next section, noting that our assumptions align closer with those of [26] where individuals carry smartphones, with the notable difference that we allow one of the individuals to not wear or carry any identifying device.

In the previous work on $SmartCondo^{TM}$project [4], they targeted the development of a method using RFID readers embedded in the environment and wearable passive RFID tags. Such an approach is limited in terms of practicality because RFID readers today—especially if endowed with large antennas to attain reasonable range—are difficult to embed in everyday surroundings without expensive retrofitting of the space (and frequently violating the aesthetics

of "home" environments). The passive RFID tags also have to be embedded in everyday items (e.g., clothing), and hence the reliability is generally compromised unless specially treated to cope with washing and other everyday wear-and-tear.

In this thesis, first, with our Estimote+WiFi method, we conducted experiments on Estimote stickers attached with simple adhesive glue on surfaces around the home which data is collected through an application running on the occupants' Android smartphones. We used an RSSI-to-distance model to convert their RSSI values to distance, fused it with WiFi-based location estimates, and used it for the localization. Based on the results of our experiments, we decided to move forward to fuse the Estimote sensors' data with PIR motion sensors (Estimote+PIR method), and focus on (a) the fusion of data collected by PIR motion sensors with data collected from tiny BLE beacons. We then (b) evaluate the effectiveness of our method through an empirical study in the $SmartCondo^{TM}$, exploring caregiver scenarios where one individual does not wear a device nor carries a smartphone, while the second (typically the caregiver) carries such a device. In addition to its applicability to realistic care-giving scenarios, the main advantage of the technique described here is that the location of the two individuals can be accurately determined [25].

This thesis is organized as follows. Chapter 2 briefly discusses the most recent related work in the field of localization and activity recognition. Chapter 3 explains our methodology and algorithm by talking about the steps we took to achieve our goal in two sections. Section 3.1 describes our Estimote+WiFi method and reports the study on using our RSSI-to-distance model on Estimote beacons data. Next, in section 3.2, the Estimote+PIR method, which is fusing Estimote beacons with PIR motion sensors, is described. Chapter 3.2.6 outlines our experimental methodology, and the results of our experiments. After testing our system and evaluating its accuracy in the real environment, we decided to make some changes to the design of the software to make it more appropriate for the real applications, these modifications are explained in detail in chapter 4. At the end, chapter 5 concludes the thesis with our findings and discusses the possible future work paths.

Chapter 2

Related work

Over the past decade, the area of indoor location estimation has resulted in many proposals and research findings, with varying degrees of applicability in real environments. We categorize these proposals by the technology they use and the main problem they are trying to solve.

2.1 RSSI-Based Methods

Within the family of RSSI-based methods, there are two different approaches. The most popular one, range-based localization, involves, first, converting RSSI values to distances via some model mapping, and then localizing the target by combining the distances to all the sensors "visible" to the target. Here, visible means sensors within a distance that allows their transmissions to be received. There are several ways to convert RSSI to distance: [23] uses path-loss modelling; [7] reports on several alternative models, including a linear model, a free-space Friis model, a flat-earth model, etc., while [32] compares distance extrapolation and interpolation, after selectively removing sensors from consideration. All these methods suffer from a certain accuracy penalty, due to the fact that the environment can substantially influence signal strength, which is not as consistent as anticipated by the RSSI-to-distance models.

The second approach is based on RSSI fingerprinting. RSSI readings for specific points in the space are collected in a database of readings at known locations; at runtime, when a new RSSI reading is received, it is compared to

the dataset and the "nearest" point (according to some definition of distance) is selected as the likely location. For example, [27] introduced a fingerprinting system utilizing passive RFID tags and four RFID readers and used a k-nearest-neighbor (kNN) method to select the nearest points to the received signal when localizing. In a building with two bedrooms and with space logically divided in a grid-like fashion of cells of size 1.5×1.5 m^2, their method achieved a reported accuracy as 96% when localizing at the granularity of a grid cell. Another fingerprinting method was described in [22] using iBeacons, for the purpose of recognizing in which room the target is located. This method has used the same BLE sensors that we are using as part of this study; However, the evaluation metric adopted in [22] is very coarse-grained, i.e., recognizing an individual's presence or not in a room, which is insufficient for ambient assisted-living services. That is why, in our study, we measure and report accuracy in terms of distance between inferred and actual locations. Furthermore, in our study, we also consider the use of PIR motion sensors as an additional source for localization. The combination of localizations based on the visible sensors has been carried out in [23] using a weighted scheme, where the weights are proportional to the corresponding RSSI. In [29], RSSI data is systematically collected at various locations, and the (unknown) target point is localized based on how close its RSSI measurements are to the profiling points. Typical of fingerprinting methods, it requires prior RSSI data collection, which is a task sensitive to the environment that needs to be repeated should the environment change in ways that impact the radio frequency propagation (e.g., when furniture is added/removed/moved).

2.2 Pedestrian-Dead-Reckoning Methods (PDR)

Another school of thought pays attention to the kinematics and relates the generation of the location estimates with the direction of movement of the individuals. An example is the pedestrian dead-reckoning (PDR) algorithm [5], where new location estimates are based on previously known locations.

For example, in [6], a PDR was proposed with WiFi and iBeacon signals, used to calibrate the drifting of the PDR algorithm by converting their RSS values to meters via a path-loss model. This family of methods requires a fall-back scheme for estimating locations in the absence of previous estimates in two cases: (a) when the initial location needs to be established, and (b) when sufficient error has accumulated, based on the estimates following the kinematics, such that a "re-initialization" of the estimation needs to take place. While we take some measures to consider the kinematic behaviour of the individuals, we do not rely on it, as the activities in which an individual is engaged in a small indoor space call for frequent changes of direction and speed, and some tasks are fundamentally unsuitable for dead-reckoning approaches (e.g., broom sweeping). In another PDR approach in [39], the authors used WiFi fingerprints with particle filtering to calibrate the PDR error after time, and they performed their experiments when the subject was walking in a path and used the location estimation approach to track the subject. In [18], an RFID-based indoor location estimation is proposed for the elderly living alone, which uses both RSSI for localizing the subject and fuses it with a PDR using accelerometer data to step and direction detection to increase the accuracy.

We hasten to add that in the IPIN 2016 offline competition [34], the best team introduced a PDR with RSS fingerprinting for the initial position with an accuracy of 2 m for a single individual in one of the spaces considered. The other four best teams performed RSSI fingerprinting, PDR, and MAC address filtering and fingerprinting with less accurate results.

In [40], PDR is used with WiFi and iBeacon fingerprinting: the iBeacon is used only where the WiFi signal is not strong enough. Similar to profiling, methods that use path-loss models rely on a model-configuration step, specific to the building and the environment where they are deployed, and changes to the environment require a re-computation of the path-loss model parameters to preserve accuracy.

In this thesis, we first started by adopting a variation of the range-based approach, via a model-fitting step, based on numerous RSSI data collected at different known distances to a beacon. The rationale is that BLE signals

are of very low power, that, especially if we ignore very small RSSI values, measurements are possible only to within a few meters from a BLE emitter. At short distances, the line-of-sight signal component is frequently the most dominant and hence the impact of the surrounding environment is lessened [24] (Estimote+WiFi method). After analyzing the results, we decided to consider only RSSI values higher (stronger) than -70 dBm (Estimote+PIR method). The way we chose this threshold is defined based on the study we did in section 3.1. In this fashion, the RSSI values only matter when they are strong enough to act as proximity sensors rather than as a model for distance calculation.

2.3 WiFi-Based Methods

In [9], the authors study the impact of multiple occupants on the RSSI measurements from WiFi access points between a pair of transmitter/receiver antennas when walking in between them. They analyzed the impact of blocking the line of sight and scattering effects on the received signal measurements to count the number of people in the area and attained an error of two people or less. In another study in [1], another multi-person localization method is proposed by studying the reflection of wireless signals off the occupants' bodies. This method, similar to the one discussed earlier, is just for detecting the number of persons in the environment and cannot identify them. Another method discussed in [35] can localize a single person with an error of only 80 centimetres on average in an area, with use of a single WiFi access point. This method calculates sub-nanosecond time-of-flight and is dependent on the use of the Intel 5300 WiFi card.

2.4 Self-Calibration Methods

A self-calibrated system is proposed in [3], where the sensors (smartphones in this case) communicate with each other to determine first their locations relative to each other, and subsequently the location of the target wearing one of these transmitters/receivers. A master node sends acoustic signals to the others to localize themselves with respect to the master node with the power

and the direction of arrival (DOA) of signals received by the two microphones on the smartphone. An iterative expectation-maximization method is then used for the nodes to communicate their local estimate to the rest of the nodes. While the reported results appear to be excellent, they are produced under a completely static network—an assumption incompatible with most realistic scenarios. Static nodes are also used in the evaluation of the method outlined in [20], which utilizes a trilateration algorithm to localize each node in a multi-sensor network after converting the received signal strengths to meters via a path-loss model. An interesting feature of this algorithm is that it incorporates a means to temporally align the collected signal strength readings. In [17], an anchor-free self-calibration is proposed by means of an "Iterative Cole Alignment" algorithm based on a spring-mass simulation [37], and ensures synchronization by assuming that all receiving devices are linked to the same laptop; this method was evaluated assuming that the target always remains within a confined area.

In general, the question of how localization methods are evaluated arises in many publications, including the ones we discussed above. For example, static node configurations and artificially confined locations for the targets are fundamentally unrealistic and are bound to fail in the scenarios motivating our research. In this study, we collect sensor data resulting from the movement of one (or two) individual(s) in an actual apartment, following real (albeit scripted for the sake of consistency) movement scenarios in this apartment.

2.5 Synchronization

Indeed, when trying to use data collected from multiple sensors for location estimation (and activity recognition), a noticeable problem is sensor synchronization: most of the time, the clocks of the emitting sensors and devices involved are not completely synchronized. The approach proposed in [31] assumes that multiple sensors—each with its own clock and all connected to a single host—collect their timestamped observations in a FIFO (First-In-First-Out) structure; the host fetches the sensor data and performs a reconstruction

of the sensor sample times, assuming a constant drift for each sensor and deterministic communication times. In our work, synchronization is not explicitly solved at the data-collection step; instead, we introduce the concept of a time "window" which abstracts the timestamp units at a coarser granularity and allows our Estimote+PIR method to ignore the imperfect synchronization of the data sources/sensors. As we will see, the window size can have a substantial effect on the accuracy of results.

2.6 Multi-Person localization

The field of multiple person location estimations has received less attention from researchers. The majority of work in this area has been limited to counting how many occupants there are in the space. For example, [2] uses only binary sensors to count the people present within an area, eliminating the outliers due to incorrect sensor firing. The algorithm is initialized with the assumption that only the minimum number of sensors are outliers, and repeatedly increases the number of outliers until a solution is produced. Unfortunately, this method cannot recognize two occupants when their movement paths cross. [12] uses RFID tags and readers for the same person-counting task: the method maintains an uncertainty circle for each person, with a radius computed as the product of their speed of movement multiplied by the time lapsed; when a new event comes from a reader, the method assumes the person most likely to have moved in the vicinity of the reader based on their uncertainty circle and their direction of movement. A more recent paper by the same group [14] uses a much more expensive Kinect sensor to actually estimate the occupants' locations.

2.7 Sensor Placement Strategies

When using motion sensors, it is important to know how to place them to achieve the best accuracy while minimizing the cost. [11] proposes a method for optimizing the placement of PIR sensors in order to meet the desired accuracy requirements while minimizing the cost. Their procedure hierarchically divides

the space to sub-areas, based on walls and other obstacles such as large static furniture pieces. It then superimposes a grid on the space, whose length is determined by the accuracy needed, and solves the optimization problem of placing sensors so that the maximum possible number of grids cells are covered. In our group's previous work, they developed a sensor-placement method that optimizes the information gain obtained by each additional PIR sensor placed in the space [36].

Chapter 3

Two Sensor-Fusion Methods

We used two approaches for indoor localization with ambient sensors. We started by examining how the Estimote stickers perform and the possibility of fusing their data with wifi-based localization, which is provided by Google FusedLocationApi, to achieve a better result. This Estimote+WiFi method was inspired by the fact that wifi is broadly available, in most residential buildings, these days and the Estimote stickers are very cheap and easy to deploy. We conducted multiple experiments to evaluate the feasibility and effectiveness of this method and realized that the Estimote RSSI values are very noisy, even when the target is not moving and in spite of using a data-smoothing method. Furthermore, a simple use of WiFi for localization using coarse-grained localization services, such as the one via Google's API, does not perform well in indoor environments. Finally, this method also suffered from a costly deployment requirement, since the RSSI-to-distance model must be re-calculated whenever the environment changes. Because of all these shortcomings, we realized that the Estimote+WiFi method was unlikely to be effective in a realistic indoor environment.

Motivated by these findings, we proceeded to develop a different approach, relying on thresholding of the Estimote RSSI values and fusion with a much more reliable sensor type, namely Passive Infrared motion sensors (PIR). The use of PIR motion sensors was motivated by the previous work that has been done around these sensors [4][36]. The intuition behind this Estimote+PIR method was to use the Estimotes to address the PIR's inability of distinguish-

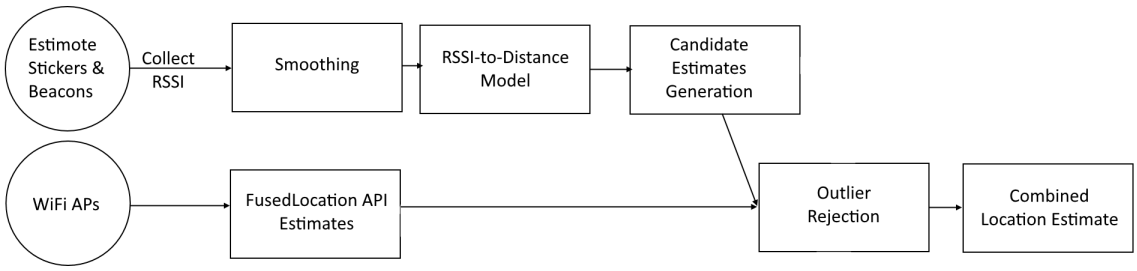


Figure 3.1: Structure of the Estimote+WiFi method

ing multiple occupants in the space. Moreover, the PIR sensors can only detect the presence of a person only when they move, hence, static occupants may not be detected. Their ability is also compromised by the need for line of sight between the sensor and the occupant so the existence of obstacles in the building may reduce the ability to detect any movement.

Section 3.1 discusses our Estimote+WiFi method and Section 3.2 discusses the Estimote+PIR method, which was validated with multiple comprehensive experiments in a real environment with one or two volunteers doing a scripted activity set in the $SmartCondo^{TM}$.

3.1 The Estimote+WiFi Method

Our localization methodology involves two deployment configuration steps. First, we *systematically* traverse the space within which localization is to be performed to *collect* RSSI *data* from each visible beacon. Next, we summarize the collected data into a *model that correlates signal strength with distance* from each beacon. At runtime, localization is performed through trilateration of the distances estimated by the model, given the run-time RSSI values observed. As an additional input, at run-time, we also consider WiFi-based localization as provided by the Google FusedLocation API. Figure 3.1 shows the different steps of this method.

3.1.1 Smoothing

To recap, our localization problem involves estimating the location of a smartphone in an indoor space covered by WiFi access points, with a number of

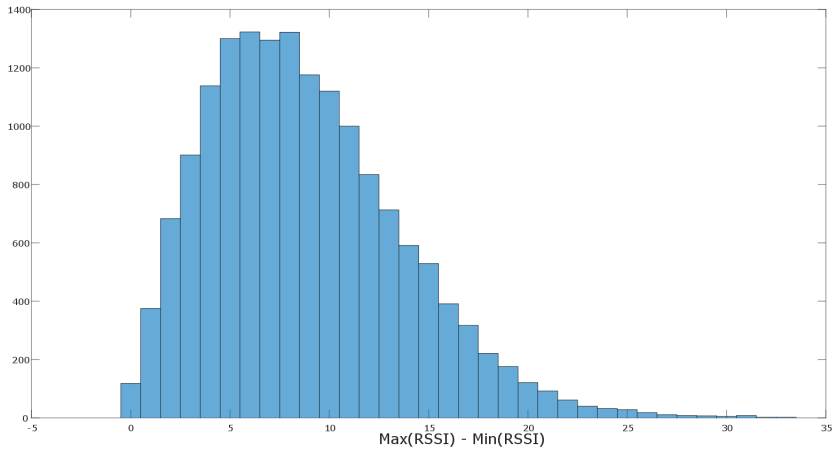


Figure 3.2: Histogram of differences between maximum and minimum values of RSSI recorded from a single sensor, at the same location, for stationary receiver at distances ranging from 1 to 10 meters away from the sensor.

Estimote stickers and beacons on the walls.

As we have already mentioned, even when the smartphone does not move and the environment does not change, RSSI values observed by the smartphone (and the corresponding location estimates) may vary. It can be confirmed by Figure 3.2, which depicts the histogram of RSSI range from a sensor received at a fixed point. This is why we decided to employ a smoothing algorithm to improve the stability of our localization process. A process that describes how to take into account the velocity and location history of the target is described in [21]. In our study, we adopt a smoothing filter motivated by the intuition that the target is not moving arbitrarily and there is a correlation between previous locations and current location. Therefore, our process smooths data with a weighted, moving-average algorithm that assigns a larger weight to the current RSSI and less to the past N estimated values, the value of N in the experiments section was set to 10. This value was determined by doing a 10-fold cross-validation over a small test set of data, selected from each experiment.

3.1.2 RSSI-to-Distance Model Construction

RSSI is the strength of the signal received by a listener from an emitting device, which in our case are Estimote stickers and beacons; based on multiple observations of correlated RSSIs at multiple locations, a model can be constructed to convert signal strength to distance. By using a single platform for the stickers/beacons, we are reasonably confident that the power of the signal emitted from each device is the same, thus avoiding the introduction of errors if multiple differently manufactured devices are used in the sticker/beacon roles. The set of distances of a smartphone equipped with a listener service from different stickers/beacons at known locations can be used to compute an estimate of the smartphone's location, via trilateration.

The RSSI is reported in dBm and, for Estimote stickers, if the transmission is set at the maximum power of +4 dBm, the RSSI was recorded to be in a range from -26 to -100 dBm. We fit a model mapping RSSI values to distance from the emitting sticker, in meters.

In order to construct an adequately accurate model, we collected around 7500 RSSI values by moving slowly, with regular steps, from a distance of 0.25m to a distance of 13m from the sticker, holding the smartphone at the same height and facing directly at the sticker. At each distance step, the average value of around 500 readings was computed as the representative RSSI value for that distance. Then we fit a model to these RSSI-distance points (shown in Figure 3.3) which was subsequently used during the localization process. Equation 3.1 shows the obtained model where $RSSI$ is in dBm and d (distance) is in meters.

$$d = 1.197 \times 10^{-4} \times e^{(-0.1307 \times RSSI)} \tag{3.1}$$

This exponential relation between RSSI and distance is expected based on the previous research on RSSI to distance models such as presented in [23] and [7].

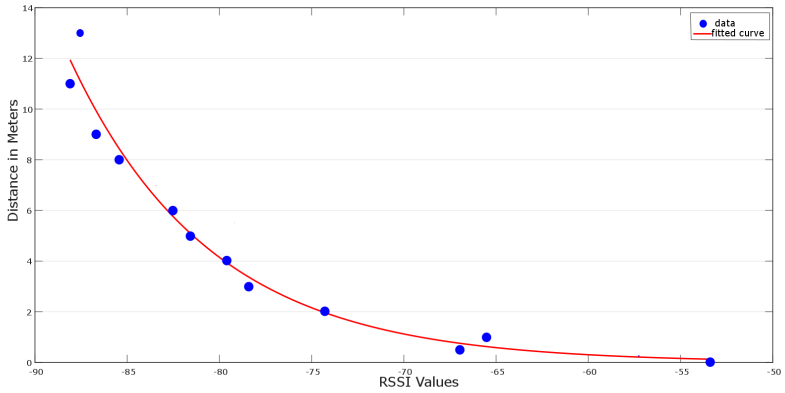


Figure 3.3: Obtained RSSI vs. distance model.

3.1.3 Generation of Candidate Estimates

Let A, B, and C be the three vertices of a triangle defined by three signal-emitting devices, visible by the signal-listener service running on the smartphone.

Using the RSSI-distance model constructed above, three values will be computed as estimates of the distance between the listener-device and each of the corresponding emitting stickers. Let these values be d_1, d_2 and d_3 respectively. Ideally, with three stickers, our target point will be at the intersection of three circles, centred at A, B and C and with the corresponding radius of d_1, d_2, and d_3. In that case, the problem can be formulated with the equation 3.2, where the stickers' positions are at $[x_i, y_i]$ while sticker i is at a distance of d_i from the to-be-localized point, at $[x, y]$. Unfortunately, because the distance estimates are not accurate, the exact intersection of the three circles may not be a single point (no single solution), or worse, such an intersection may not even exist.

A typical method for addressing the problem of inaccurate distance estimates involves using Equation 3.2 to formulate an LSE (Least Squared Error) instance [29], [30]. With the LSE approach, the result of Equation 3.2 can be determined using equation 3.3, where A and b hold the equations 3.4 and 3.5.

$$\begin{cases} (x_1 - x)^2 + (y_1 - y)^2 = d_1^2 \\ (x_2 - x)^2 + (y_2 - y)^2 = d_2^2 \\ (x_3 - x)^2 + (y_3 - y)^2 = d_3^2 \end{cases} \tag{3.2}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = (A^T A)^{-1} A^T b \tag{3.3}$$

$$A = 2 \begin{bmatrix} x_3 - x_1 & y_3 - y_1 \\ x_3 - x_2 & y_3 - y_2 \end{bmatrix} \tag{3.4}$$

$$b = \begin{bmatrix} (d_1^2 - d_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) \\ (d_2^2 - d_3^2) - (x_2^2 - x_3^2) - (y_2^2 - y_3^2) \end{bmatrix} \tag{3.5}$$

A further, combinatorial, step of processing is implemented on top of the trilateration and is described in subsection 3.1.5.

3.1.4 WiFi-based Location Estimation

The FusedLocation API provided by Google estimates the location of a smartphone with help of GPS, WiFi and cellular access points. With the use of a variable called *priority*, we can force it to use only WiFi because our interest here is indoor localization, and GPS is not useful in this context. To obtain data from this service, one only need is a smartphone with the application installed and WiFi enabled; then, when the phone moves within an area covered by WiFi access points, the application estimates, in real time, the location of the smartphone.

3.1.5 Outlier Rejection

A particular feature of our approach is the way in which outliers are rejected. Instead of qualifying a particular measurement as an outlier, we systematically eliminate candidate localizations as outliers. Specifically, for each location, we produce multiple candidate localizations. It is the clustering of the candidate locations that decides which candidate localizations will be called outliers. The mix of candidate estimations of a target's location is enriched with the introduction of the FusedLocation-based estimate. Hence, the FusedLocation

estimate is of value only if, and to the extent that, it is not an outlier localization.

The question of generating multiple candidate localizations is addressed by performing multiple trilaterations. Specifically, the trilateration LSE is applied to *every* three-sticker/beacon combination, thus producing a corresponding plurality of estimates for the location of the smartphone. Whereas it would have been possible to run LSE for all the received beacon signals (i.e., a single multilateration), we instead run as many LSE sub-problems as allowed by the combinations of, three at a time, measurements of the signal from the beacons/stickers. Essentially, if a beacon's measurement is an outlier, its impact on the localizations where it participates may be to cause an outlier localization. Instead of characterizing the quality of the data measurement as an outlier, we characterize as outlier its impact on localization (if it is indeed hurtful to the localization). In general, such outliers are to be expected as the impact of nearby structures on signal strength, objects between the device and beacon (e.g. person's body), power issues, and etc.

In our study, we identify and eliminate two types of outliers. First, we ignore all location estimates that fall outside the experiment area, because we assume that with use of other services, such as ingress/egress monitoring to an assisted living facility, applications will be aware whether we are inside or outside the area in which localization is performed. Second, we use a K-Nearest-Neighbor (KNN) clustering algorithm to exclude "unlikely" localization candidates. Intuitively, our algorithm aims at identifying the densest cluster of location estimates, and, excludes all other location estimates as noisy outliers. The value of K is determined by using a cross-validation approach to find out the K that results in the best accuracy for the whole localization process in a sample experiment.

3.1.6 Location-Estimate Fusion

Once all possible trilateration-based location estimates and the FusedLocation estimate have been collected and the outliers have been eliminated, our localization process produces a weighted average across all of them. The weights,

α_i, are selected such that estimates that are distant from the rest are given less weight. Specifically, α_i is the weight of the i–th estimate (corresponding to estimate point x_i) and is calculated as the inverse of the summation of its distance to all remaining estimates. Equation 3.6 captures how the weight for the i–th estimate is calculated. Note that, here, x_i's are two-dimensional vectors, and the distance function is the ordinary Euclidean distance.

$$\alpha_i = \frac{1}{\sum_{j=1}^{n} distance(x_i, x_j)} \tag{3.6}$$

The combined location estimate (equation 3.7) is calculated as a weighted and normalized sum of all the individual estimates, i.e.,

$$\frac{\sum_{i=1}^{n} x_i \alpha_i}{\sum_{i=1}^{n} \alpha_i} \tag{3.7}$$

Roughly speaking, this choice for weighted average tends to place the localized point in the "middle" of the densest area of individual estimates. Keeping in mind that these estimates are the result of combinations of trilaterations (plus the WiFi localization), it essentially places the final location within a small distance away from that which would approximately satisfy most of the produced trilateration estimates. This, however, does not mean that the localization is exact in the absolute sense, hence we conduct a number of experiments to assess its accuracy.

3.1.7 Evaluation and Results

We developed two mobile applications, one for Android and a second for iOS devices. The applications are in effect simple background services that listen to packets from the Estimote nodes and save the RSSI values of the transmissions received, together with their corresponding timestamps. In parallel, the service also invokes the Google FusedLocation API to obtain the phone's latitude and longitude at each time stamp. The localization method, described in Section 3.1.1 through 3.1.6, is implemented in MATLAB on a remote server, i.e., not on the smartphone devices. The localization processing time averaged approximately 30 milliseconds, which implies that it is feasible to perform it as a service for real-time localization.

We conducted three experiments, in three different buildings and in three different spaces, which we describe in detail in this section. In the first experiment, taking place in the building where the Computing Science department is housed, we used both BLE sensors and Google FusedLocation API to localize the phone; because we had an adequate number of WiFi access points for the WiFi localization (in contrast to the relative paucity of WiFi access points in the remaining two buildings). In the second and third one, only the BLE sensors were used.

Experiment 1

We ran an experiment in an area in Computing Science department, at the University of Alberta, consisting of three corridors with lengths of 10.5, 54.5 and 19.5 meters and width of 1.5 meter. There were several WiFi access points in the area and 26 beacons were affixed on the walls and the experimenter moved around the area with a smartphone (Samsung Galaxy S4) in her hand, maintaining the smartphone at a consistent height in order to reduce the effect of environment on the received signals.

Figure 3.4 shows the map of our experiment area. The brown stars correspond to the stickers on the walls, 26 in total. The experimenter walked through the area starting from the bottom-left corner, staying at each of the blue points for 30 seconds, and then moving to the next one.

Experiment 2

In this experiment, we examined the accuracy of indoor localization with just the BLE sensors. This experiment took place in an empty room, at the Edmonton Clinic Health Academy (ECHA) of the University of Alberta. The room's dimensions are 10.73 meters by 10.9 meters and it was populated by 42 stickers and beacons fixed to the walls and the same experimenter was moving around with the same smartphone in her hand. We tried to mimic in this experiment as faithfully as possible the conditions of the first (corridor) experiment; hence, the experimenter walked through the area starting from the bottom-left corner and stopped at each blue circles for 30 seconds and then

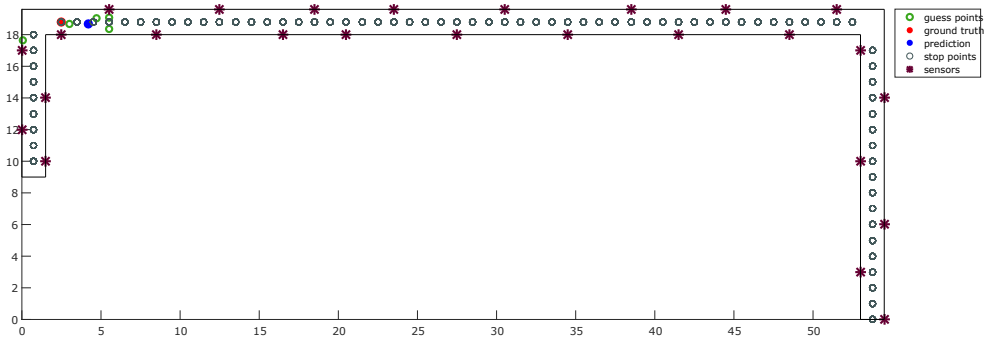


Figure 3.4: Experiment 1: Sensor deployment and example location estimates,

moved along. Figure 3.5 shows the map of the space. The light-blue circle points are the locations where she stopped and gathers data on her phone for 30 seconds. We had 100 stop points, 1 meter apart from each other.

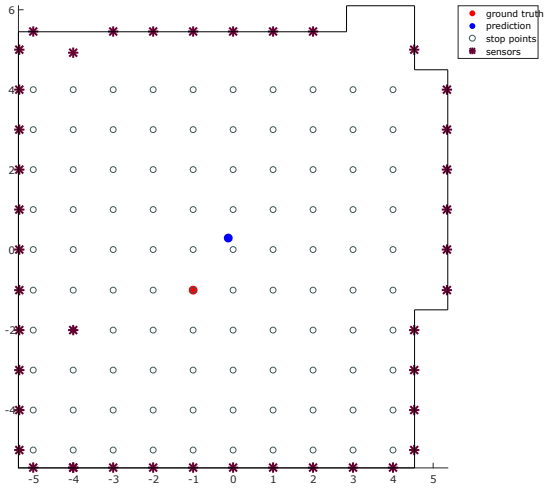


Figure 3.5: Experiment 2: Sensor deployment and example location estimates,

Experiment 3

The last experiment was conducted in the living room of a small house in a residential area, around 4.3 by 8.3 meters. The map of the space is shown in the Figure 3.6. Just like the previous experiments, the experimenter was moving around with a Galaxy S4 in her hand, starting from point number 1, stopping for 30 seconds at each point and then moving on to the next point. 10 brown stars on the map show 10 BLE sticker which were glued to the wall and the green circles show the possible location estimations coming from the localization algorithm, that help to predict the final location as explained in the sections 3.1.1 to 3.1.6.

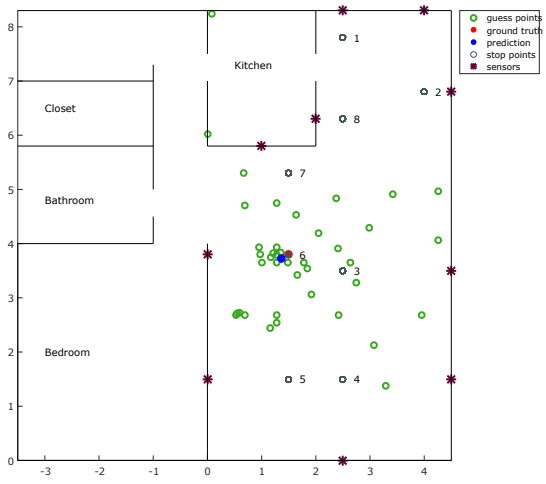


Figure 3.6: Experiment 3: Sensor deployment and example location estimates.

Results and Discussion

The results of the three experiments are summarized in Table 3.1. We were quite disappointed to discover that no useful location estimate was obtained through the FusedLocation API - in almost all cases, the received estimates fell well outside the experiment area. Hence, when we report WiFi accuracy in Table 3.1, we considered every point that WiFi gives us. But, after fusing all the estimates, because almost all of the WiFi predictions were out of the area, WiFi did not make any difference to the resulting accuracy. This reveals

the fact that simple use of WiFi for localization such as the one via Google's API does not perform well in indoor environments.

Table 3.1: Results (in meters).

Experiment	**Devices**	**Avg. Error**
Experiment 1	BLE stickers/beacons	8.45
Experiment 1	WiFi via FusedApi	30.90
Experiment 1	Combination of WiFi and BLE	8.45
Experiment 2	BLE stickers/beacons	2.32
Experiment 3	BLE stickers/beacons	1.80

The poor result of experiment 1 with respect to experiment 2 and 3 makes more sense when we realize that the corridor environment is a long and narrow area and because our algorithm needs at least 3 sensor points to apply the trilateration, there will be very few sets of useful sensors for localizing the target. In most cases, the subject is far from most of the BLE transmitters. On the other hand, in both experiments 2 and 3, we have a wider area, resulting in having more sets of 3 sensors forming a triangle for producing location estimates. Another reason for the variability in the accuracies reported in Table 3.1 can be the difference in the building construction, which can affect the signal strength. The building in the second experiment uses a significant amount of structural steel beams, whereas the residential space is almost entirely composed of non-metallic structure elements.

While it is true that we did not use a profiling approach for WiFi, as was used for BLE, it is clear that the deployment of multiple WiFi access points for the purpose of location estimations is not an economically sound approach when inexpensive BLE devices abound.

As mentioned before, our experimentation with the Estimote+WiFi method shows that it is not reliable for realistic environments and scenarios.

- Even though our experiment was conducted in a very sparse area and the subject was moving artificially slowly, pausing in front of the Estimotes with the smartphone in direct line-of-sight to the Estimotes, the resulting

accuracy was relatively poor. This accuracy is bound to deteriorate if the subject was to move normally in a typical home environment.

- Furthermore, this method relies on a rather stringent assumption of every subject carrying a smartphone. This assumption is unrealistic especially when the system is going to be used by elderly people, who may forget to always carry their smartphone.

These observations led us to a new idea of changing the way we use Estimotes' data, so that we don't convert RSSI values to distance anymore. We also decided to fuse Estimote sensor's data with more reliable sensors like PIRs to create a better system. This new method is explained in detail in section 3.2

3.2 The Estimote+PIR Method

Based on the experiments reported in the previous section on Estimote+WiFi method, we realized that Google fused location API is not reliable inside buildings and Estimote stickers data can be very noisy especially when the occupant is relatively far from the Estimote and the RSSI value is less than -70dBm. Here in this section, we describe the Estimote+PIR method we developed to alleviate these problems, for indoor localization in the $SmartCondo^{TM}$.

Figure 3.7 provides the logical view of the Estimote+PIR method architecture. The diagram depicts the two independent sensor-data collection paths combined at the server. Note that the architecture could technically admit more such independent simultaneously operating sensor feeds. The upper-left branch (Estimotes) captures eponymous data collection carried out by the smartphone device(s), and, in the future, by wearable devices. Estimote beacons are attached to objects in the surrounding space, with a considerable number of them attached to static objects (e.g., walls), or objects with trajectories known in advance (e.g., doors). Estimote "stickers" are fairly small and do not greatly impact the look-and-feel of the space; their interesting shapes and colours could even allow them to be perceived as decorative elements. The collection of data (RSSI values) is performed by Android devices

running a special-purpose application which is aware of all installed stickers and their locations. When the device (smartphone or wearable), comes to the vicinity of any of these stickers, the application recognizes their presence and collects information about their RSSI and accelerometer signals. The RSSI is reported in dBm, ranging from -26 to -100 dBm when the transmitting power is set to a maximum of +4 dBm. The Android application streams this data to the $SmartCondo^{TM}$ server every second. The format of the data sent from the Android device to the server is $< t_i, device_{ID}, beacon_{ID}, RSSI >$, implying that at the specific timestamp t_i, the person carrying the device with ID=$device_{ID}$ received a transmission with a strength of $RSSI$ from the Estimote with $beacon_{ID}$. Henceforth, we are using the terms $device_{ID}$ and p_{ID} interchangeably. Figure 3.8 shows the logical view of the data gathering process in the $SmartCondo^{TM}$for the Estimote+PIR method.

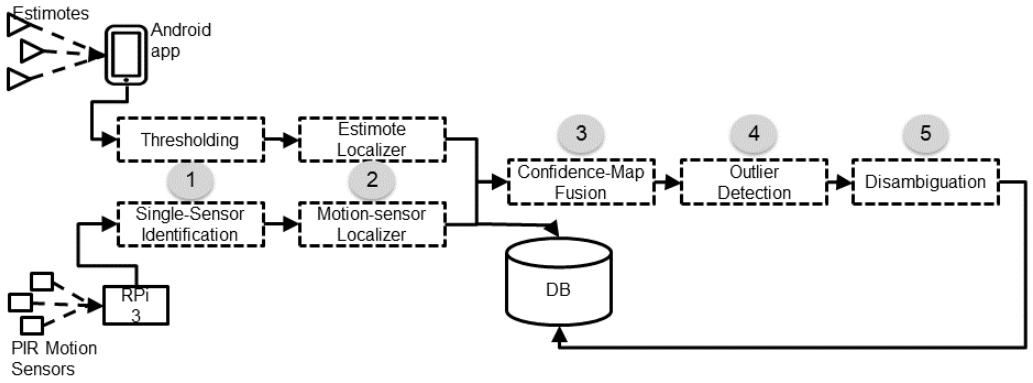


Figure 3.7: Structure of the Estimote+PIR method. PIR: Pyroelectric ("Passive") Infrared. DB: Data Base. RPi: Raspberry Pi 3.

The lower branch captures the anonymous sensing carried out by the PIR spot-type motion sensors placed on the ceiling. The PIRs can detect any movement within their sensing area which is a diamond-shaped area, with the two diagonals equal to approximately 1.75 and 2 meters. Groups of up to three motion sensors are connected via wires to a nearby wireless node, running a purpose-built firmware on a Texas Instruments MSP430 microcontroller using TI's proprietary wireless module (CC1100). These nodes operate in the

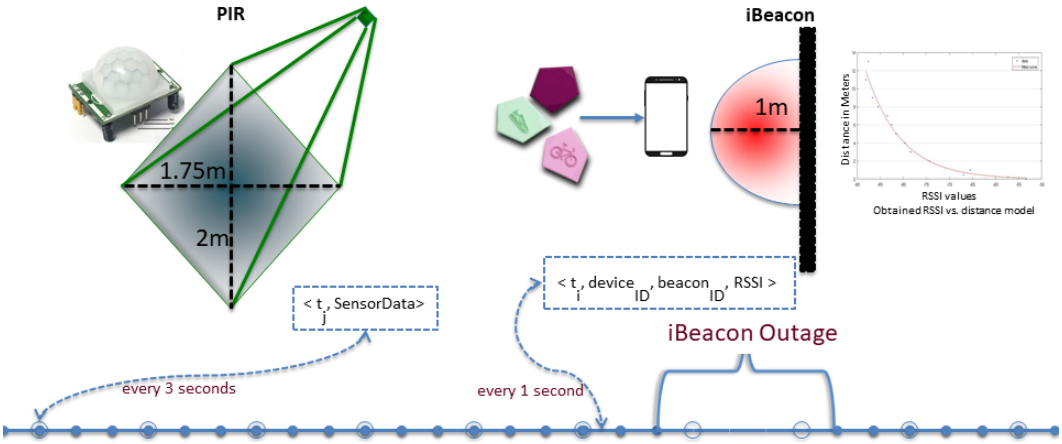


Figure 3.8: The data gathering logic in Estimote+PIR method. Motion sensors sense data within the diamond area their facing. Estimote sensors send RSSI values and by applying a threshold of -70 dBm, they sense objects within 1 meter distance to themselves. This threshold comes from an RSSI to distance model that we calculated in section 3.1

900 MHz band, thus avoiding the heavily utilized 2.4 GHz band. The nodes wirelessly transmit the sensor observations to a Raspberry Pi 3 (RPi 3) with Internet access, which in turn uploads the data to a cloud-based server every three seconds. The format of the data uploaded by the RPi 3 to the server is $< t_i, data >$, where the "*data*" element is a bitmap equal in length to the number of motion sensors installed. A 1 (0) at the i_{th} position of the data bitmap implies that the sensor, corresponding to the i_{th} index, detected (did not detect) movement within its corresponding sensing area. From a practical perspective, we should mention that in our three installations to date we have been able to hide the wires and the nodes inside ceiling tiles and behind cabinets, in order to minimize their impact on the aesthetics of the home.

It is important to note here some interesting similarities and differences between the two types of sensor data. Both types of data elements are timestamped with the time of the emitting device: the Android smartphone in the case of Estimotes, and the Raspberry Pi in the case of the motion sensors. Both include a payload: the $< beacon_{ID}, RSSI >$ tuple in the case of Estimotes, and the *data* element in the case of the motion sensors. Note that the former includes information about a single beacon while the latter

composes information about all the deployed motion sensors encoded in a bitmap. The most interesting difference between the two is the fact that Estimote data-transmission events are eponymous: each event includes the ID of a person, p_{ID}, (carrying the corresponding device, $device_{ID}$) perceived by the firing $beacon_{ID}$. This important difference characterizes motion sensors as anonymous and Estimotes as eponymous.

Our Estimote+PIR method involves five steps, diagrammatically depicted as a processing pipeline in Figure 3.7. The first two are specific to each type of sensor, and focus on data pre-processing and the generation of a first location estimate based only on the sensor(s) of this type. Should a new sensor type be integrated into our infrastructure, a corresponding step sequence would have to be developed specifically for this new sensor type. The remaining three steps are general and focus on fusing the sensor-specific location estimates.

3.2.1 Data-Stream Pre-Processing

As shown in Figure 3.7, all "raw" data is pushed to the database. A first, pre-processing step is applied to the raw data and the results are also stored in the database. As we have already discussed, each type of data is pre-processed differently. *RSSI thresholding* is applied to the Estimote data stream: a minimum threshold of -70 dBm is used to select the RSSI readings that are "significant" enough to be used for location estimation. This specific threshold value was motivated by experiments reported in Section 3.1 and Figure 3.3: roughly speaking, -70 dBm (or higher) RSSI strength suggests that the device is within approximately one meter of the transmitting beacon. In this fashion, the RSSI sensing effectively becomes an eponymous *proximity* sensor. The motion-sensor bitmaps are pre-processed to separate the individual motion sensor events. An additional pre-processing step is applied at this stage to add information helpful to the semantics of subsequent activity recognition, such as to label certain events as related to specific "actions" (e.g., an Estimote beacon attached to a kettle is associated with the action of "cooking").

The pre-processed data streams are then fed to the type-specific localizers. We recognize that given the various technologies that might co-exist, a

general format for the localizers needs to be defined to address current and future demands, and yet be able to integrate into the system easily. To this end, Figure 3.9 shows the UML (Unified Modeling Language) design of the sensor, event, and localizer classes, described in the following subsections. Figure 3.9 shows the base *Localizer* class with abstract methods for initializing its sensors and handling their corresponding incoming events. These methods are implemented in the children classes (i.e., the $MotionLocalizer$ and $EstimoteLocalizer$), since each sensor type demands a different localization algorithm. The *Sensor* class has an ID which is a common field for every sensor. In addition, the $EstimoteSensor$ and $MotionSensor$ classes have an associated *location* where they are installed, and a *sensing_polygon* defining the area within which they perceive movement. They implement the *can_see* method, which decides whether a particular location is covered by their sensing area. The $MotionEvent$ and $EstimoteEvent$ classes are, children of the *Event* class, and have a *timestamp* property. In the future, to add a new sensor type, X, to the system, one would first have to add a set of corresponding $XEvent$, $XSensor$, and $XLocalizer$ classes, inheriting from *Event*, *Sensor*, and *Localizer*, respectively. The new $XLocalizer$ class would also have to be added to the list of a system's localizers to process incoming $XSensor$ events and feed a corresponding output location estimate to the subsequent fusion steps.

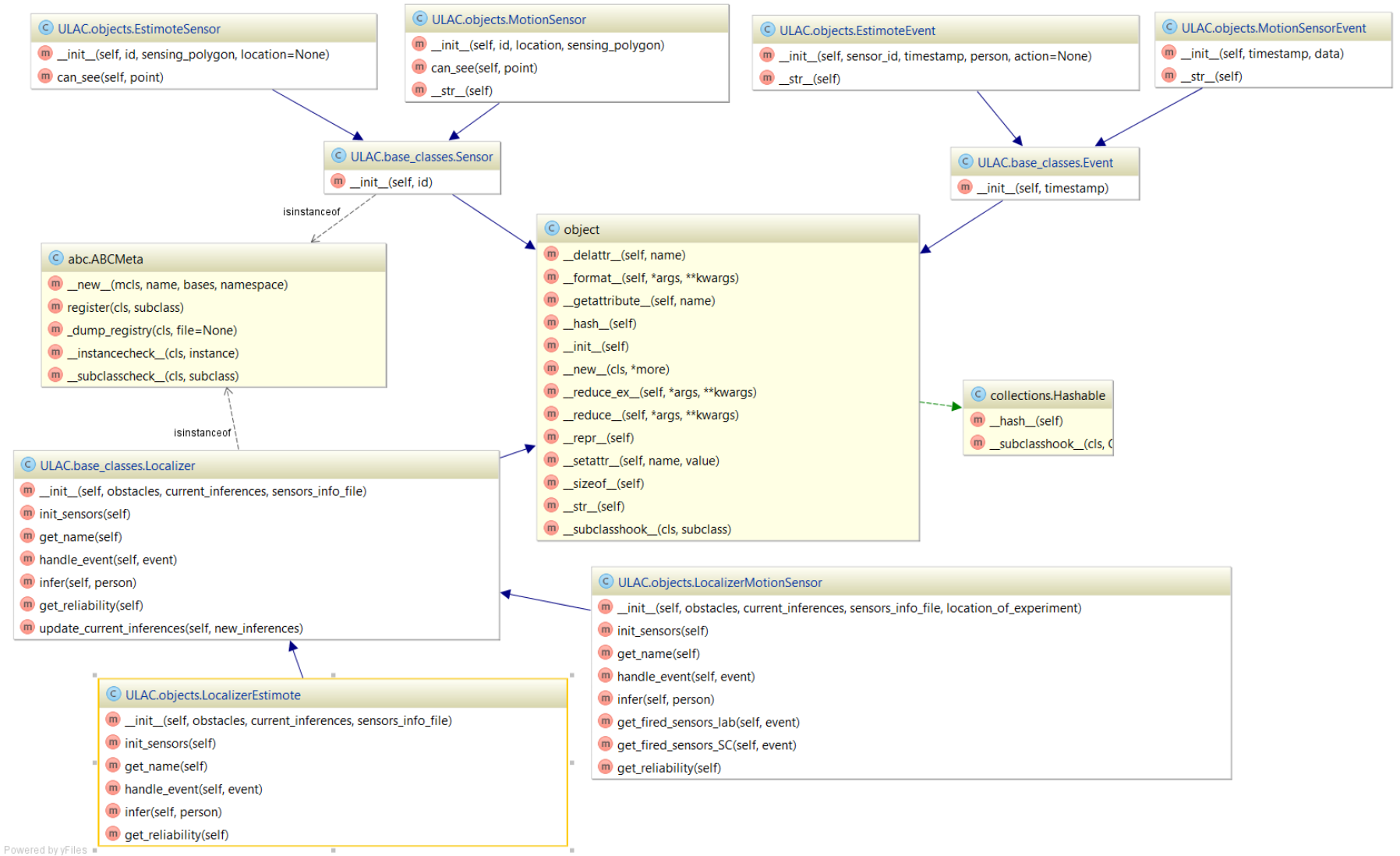


Figure 3.9: UML (Unified Modeling Language) diagram of the basic objects in Estimote+PIR method.

3.2.2 Sensor-Specific Location Estimation

Algorithm 1 Incoming event processing

1: **procedure** SensorFireHandler(e_i)
2: **if** $e_i.type = ESTIMOTE$ **then**
3: **if** $e_i.RSSI > -70$ **then** ▷ //Pre-processing part for Estimote events
4: $confMap^{(L_{est})}_{t_i,p_{ID}} = EstimoteLocalizer(e^{(L_{est})}_{i,p_{ID}})$
5: **for** p_{ID} in P **do** ▷ //P is a set of all persons in system
6: $motionMap = confMap^{(L_{ms})}_{t_{i-1},p_{ID}}$ ▷ //Most recent confMap from L_{ms}
7: **if** $e_i.person = p$ **then**
8: $estimoteMap = confMap^{(L_{est})}_{t_i,p_{ID}}$
9: **else**
10: $estimoteMap = confMap^{(L_{est})}_{t_{i-1},p_{ID}}$ ▷ //Most recent confMap from L_{est}
11: **end if**
12: $confMap_{t_i,p_{ID}} = Fuse(motionMap, estimoteMap, t_i, p_{ID})$
13: **end for**
14: **end if**
15: **end if**
16: **if** $e_i.type = MOTION$ **then**
17: $e_i^{(L_{ms})} = preProcess^{(L_{ms})}(e_i)$ ▷ //$e_i^{(L_{ms})}$ has a set of of fired sensors
18: **for** p_{ID} in P **do**
19: $confMap^{(L_{ms})}_{t_i,p_{ID}} = motionLocalizer(e_i^{(L_{ms})})$
20: $motionMap = confMap^{(L_{ms})}_{t_i,p_{ID}}$
21: $estimoteMap = confMap^{(L_{est})}_{t_{i-1},p_{ID}}$
22: $confMap_{t_i,p_{ID}} = Fuse(motionMap, estimoteMap)$ ▷ //Produce final confMap for p_{ID}
23: **end for**
24: **end if**
25: $PostProcess()$ ▷ //Disambiguate for all persons in system
26: **end procedure**

For the Estimote+PIR method on $SmartCondo^{TM}$, we have adopted the methodological assumption that individual localizers should be developed to work with each specific data stream producing a location estimate. Our location-estimate synthesis algorithm is general enough to take into account any number of such estimates and synthesize them into an overall location estimate. This design decision enables the extendibility of our platform: a new sensor-data stream implies the need for a new localization algorithm whose

output is a location estimate that can be fed into the existing synthesis step. In this study, an Estimote-based Localizer and a Motion-Sensor-based (anonymous) Localizer are described.

As shown in Figure 3.9, every localizer should implement the same set of behaviours, including (a) initialization of the sensors whose data it consumes and (b) handling a data-transmission event. The key characteristic shared by all localizers is that their output should be described as a function of a "confidence" (or probability) metric over the space, which is represented as a two-dimensional grid of locations. That is, each localizer L_x produces as output a $< t_i, m_{xy}^{(L_x)}, p_{ID} >$ tuple. The $m_{xy}^{(L_x)}$ is a positive metric (the larger the value, the more confident the localizer is about its estimate) that the individual p_{ID} is at location (x, y) at time t_i. If a localizer is anonymous, the output is independent of p_{ID} (i.e., the same for any candidate p_{ID}).

Algorithm 1 describes how each new event is processed upon arrival. The term "confMap" *confidence map* refers to the location estimates produced by each individual localizer, and by the subsequent fusion step. A number of *confidence maps* are illustrated in Figure 3.10. A localizer relying on anonymous sensors (e.g., motion sensors) produces a single confidence map, which is akin to a heat map of the space, with the color of each location corresponding to the localizer's confidence that "some" occupant—any one of the known occupants in the space—is in that location. A localizer relying on eponymous sensors (e.g., Estimotes) produces a set of confidence maps, each one corresponding to an individual occupant, with a specific p_{ID}.

The Estimote Localizer produces the confidence map according to Algorithm 2. t_i and p_{ID} denote the timestamp of the incoming sensor event ($e_{i,p_{ID}}^{(L_{est})}$) and the person whose movement caused the event. As mentioned above, the sensing area of the Estimote beacons and stickers, given the -70 dBm threshold, is approximated by a circle of diameter 1 m around each Estimote. The motion-sensor localizer generates a confidence map as described in Algorithm 3 every time it receives a new motion-sensor event, $e_i^{(L_{ms})}$, containing information about all the motion sensors that fired in that timestamp.

Algorithm 2 Estimote Localizer

procedure EstimoteLocalizer($e_{i,p_{ID}}^{(L_{est})}$)
  for all (x,y) **do** ▷ // All (x, y) in the area map
    if $(x, y) \in SensingArea(e_{i,p_{ID}}^{(L_{est})}.sensor_{beacon\_ID})$ **then** ▷ //If sensor in the event sees (x, y)
      $m_{x,y} = 1$
    else
      $m_{x,y} = 0$
    end if
    $confMap_{t_i,p_{ID}}^{(L_{est})}(x, y) = m_{x,y}$ ▷ //Set confidence value for point (x, y)
  end for
return $confMap_{t,p}^{(L_{est})}$
end procedure

3.2.3 Location-Estimate Fusion

Depending on the algorithm it uses to compute its corresponding confidence-map for its location estimate, each localizer may use a different range of values in the representation of their confidence. In order to construct a common confidence value across all (two, in the case of this experiment) contributing localizers, the third step in the process involves a weighted summation of the input confidence maps, corresponding to each p_{ID}. When no eponymous sensors have been deployed in the space, the scheme reverts to a single confidence map that recognizes the likely locations for all individuals, without distinguishing among them.

The fusion step is simply the summation of the confidence values across the entire x, y-space using a weighted sum. Anonymous values are added to any eponymous values, but eponymous values can only be combined with the corresponding eponymous values (i.e., values for the same person p_{ID}). The weights are described in 4 as $EstimoteReliability$ and $MotionReliability$. In this study, their values were set to 0.7 and 0.9, respectively; these values are based on the observed accuracy of the Estimote and motion sensors in previous studies in [24], [36], which revealed that PIR motion sensors can be more accurate than Estimotes.

Algorithm 3 Motion Localizer

```
1: procedure MotionLocalizer(e_i^(L_ms))
2:     for all (x,y) do                          ▷ // All (x, y) in the area map
3:         for all sensor in firedSensorSet do ▷ firedSensorSet is inside e_i^(L_ms)
4:             if (x, y) ∈ SensingArea(sensor) then
5:                 m_{x,y} = 1
6:             else
7:                 m_{x,y} = 0
8:             end if
9:         end for
10:        confMap_{t_i}^{Motion}(x, y) = m_{x,y}      ▷ //Set confidence value for point
   (x, y)
11:    end for
12: return confMap_{t_i}^{Motion}
13: end procedure
```

3.2.4 Reasoning about Outages and Outliers

As discussed above, the purpose of the fusion step is to merge the most recent location estimates produced by each distinct localizer into a single location estimate. However, either due to sensor malfunction or channel interference, one of the localizers may experience an "outage" (i.e., it may be silent for a long time). This case is depicted in Figure 3.8. In this case, as new location estimates are produced by the other localizers, they will have to be fused with ever-older estimates. In this case, Estimote+PIR method applies a confidence-reduction penalty to the fused estimate in order to recognize the fact that the two sources of evidence are out-of-sync and they may represent different states in the real world.

The function $ReduceConfidence$ in Algorithm 4 receives as input the set of latest confidence maps from all localizers, and computes the time lapsed between the oldest and the most recent, $t_{diff} = t_{max} - t_{min}$. It returns as output the value of $\frac{1}{t_{dif}+0.2}$ as the confidence penalty, which is applied as a multiplier to each location in the fused confidence map. This value is selected so that older confidence maps will be penalized more.

Algorithm 4 Location-Estimate Fusion

1: **procedure** FUSE(motionMap, estimoteMap, t_i, p_{ID})
2: $ER = EstimoteReliability$
3: $MR = MotionReliability$
4: $confMap = create\ empty\ map$
5: $[confidencePenalty^{(L_{est})}, confidencePenalty^{(L_{ms})}] = ReduceConfidence(estimoteMap, motionMap)$ ▷ //The above line calculate s confidence penalty for confMaps of L_{est} and L_{est}
6: $lastLocationEstimate = getLastLocationEsimate(p_{ID})$ ▷ //Last location estimate for p_{ID}
7: **for** $all(x, y)$ **do**
8: $estimoteConf = m_{t_i,p_{ID}}^{(L_{est})}(x, y) * ER$ ▷ // Multiply weights to $confMap^{(L_{est})}$
9: $motionConf = m_{t_i,p_{ID}}^{(L_{ms})}(x, y) * MR$ ▷ // Multiply weights to $confMap^{(L_{ms})}$
10: $confMap_{t_i,p_{ID}}(x, y) = estimoteConf + motionConf$ ▷ //Sum the confidences
11: $distance = A^*.FindPathLength(lastLocationEstimate.location, (x, y))$ ▷ //Find distance
12: $speed = \frac{distance}{t_i - lastLocationEstimate.time}$ ▷ //Calculate speed
13: **if** $speed > speedLimit$ **then**
14: $confMap_{t_i,p_{ID}}(x, y) *= speedPenalty$ ▷ //Multiply speed penalty
15: **end if**
16: **end for**
17: **return** $confMap_{t_i,p_{ID}}$
18: **end procedure**

Next, the method determines if the displacement by which the individual has potentially moved—from the last timestamp to the current one—is a potential "outlier". This information is important for the purpose of rejecting the result of misfiring sensors that would result in placing the person at an unlikely distance from the previous location estimate. Let us call the weighted sum for person p_{ID} at time t_i as $s_{p_{ID},x,y}(t_i)$; then, the output location estimate (x, y) for p_{ID} is the average of $\arg\max_{x,y \in Grid} s_{p_{ID},x,y}(t_i)$, where $Grid$ are the square blocks in the area map.

The distance between successive location estimates is calculated with the help of a A^* algorithm [16], using a finer location grid than the grid used for localization purposes. In the current configuration of our Estimote+PIR

method, the A^* algorithm runs on a 0.2 m grid, compared to the localization process which assumes a 1 m grid. The A^* grid honours the spatial constraints imposed by the obstacles (i.e., not going through walls). The choice of a fine grid for the A^* search is motivated by the need to capture features of the space that might hinder the movement of an individual. Using A^* is motivated by our preference to include search algorithms that could work in a dynamic environment. Realistically, individuals and (some) obstacles may be moved (and tracked); therefore, spaces are dynamic enough to preclude the use of static shortest-path algorithms. A person's potential speed is calculated based on the distance between the current and last location estimates and the time-difference between them; if the calculated speed is more than the normal speed of a walking person (approximately 1.3 m/s), the confidence of the cells in the new location estimate is reduced by a factor of 0.5; these steps are incorporated in Algorithm 4. The A^* algorithm starts from the previous location estimate $(x, y)_{i-1,p_{ID}}$ at time t_{i-1} and searches for the shortest path to the current location estimate$(x, y)_{i,p_{ID}}$. To speed up the process, at each point po_{middle} in the middle of the search, as soon as the length of $path_{(x,y)_{i-1,p_{ID}} \rightarrow po_{middle}}$ plus the Euclidean distance between po_{middle} and $(x, y)_{i,p_{ID}}$ is more than $normal\ speed \times (t_i - t_{i-1})$, the search stops and the confidence map is penalized. These steps are shown in lines 12–16 in Algorithm 4.

3.2.5 Disambiguation of Anonymous Persons

Let us consider the case where only motion sensors are utilized; in that case, the confidence maps produced will contain areas where the motion-sensor localizer identifies the potential presence of "some" individuals. In principle, the number for these areas will be less than or equal to the number of persons in the space: if two or more people congregate in roughly the same location, then there will be a single area corresponding to their presence as a group. This is exactly the scenario investigated in [4].

If all individuals are also identified through an additional technology (such as in the case where all individuals are carrying smartphones and are recognized in the vicinity of Estimotes), then the sensor-fusion step results in

merging the evidence collected from the various sensors in a single confidence map, where all areas are annotated with a person p_{ID} to indicate some confidence in the presence of this specific person in the area.

There is yet another scenario: when one of the occupants is not tracked by anything other than the motion sensors; this situation may occur either because of an outage in Estimote data-transmission events, or because the occupant is not carrying any smartphone at all. This case happens in Figure 3.10, where person 1 carries a smartphone and is associated with the confidence map of Figure 3.10c, while person 2 is localized only by motion sensors and corresponds to the confidence map of Figure 3.10d. Then, it is possible to disambiguate the "anonymous" occupant (person 2 in Figure 3.10) with a post-processing step given that we have estimated where the other participant is located. This process results in Figure 3.10f for the second participant. This step involves a Gaussian mixture model (GMM) that treats the normalized confidence values of the confidence map as probabilities and clusters them. The GMM is not provided with any information about the number of individuals present, and attempts to fit the best model possible [13]. In our example, in Figure 3.10d, the GMM returns two clusters, corresponding to the two dark red areas. Then, the confidence of the points in the cluster that has a distance smaller than 0.5 m to the areas that have been annotated with the p_{ID}s of the smartphone-carrying individuals (Figure 3.10c) is reduced, hence "subtracting" from the confidence values, and the remaining cluster in the confidence map, corresponding to the anonymous person (person 2) will have a higher probability of them being there, resulting in the confidence map at Figure 3.10f for person 2. This process is performed as the very last step of Algorithm 1.

3.2.6 Evaluation and Results

Twenty-six participants were recruited to spend one two-hour shift—either alone or in pairs (seven pairs)—in the $SmartCondo^{TM}$. The participants were asked to follow a scripted sequence of activities (i.e., an activity protocol). This protocol started with the subjects placing their personal belongings in

the entrance closet; followed by performing some exercises in front of a Kinect; simulating personal-care activities including toileting and bathing; preparing a meal, eating it, and cleaning up; simulating doing laundry; playing some games on a tablet, and watching TV. Some activities were simulated (e.g., personal care, dressing) and others were real (e.g., cooking, ironing, exercising). For the two-participant sessions, the protocol was the same for both subjects, with the exception that the order of the activities was slightly modified, and that both participants were involved in the meal preparation and TV-watching activities. Each of the activities in the protocol was scripted in details as a sequence of smaller tasks. For example, the instructions for the meal-preparation activity were to get the frying pan from the cabinet, bring eggs from the fridge, get a spoon, stand in front of the kitchen island, cook scrambled eggs, etc. A tablet was provided to each participant, running an application that prompted them to perform the next step; when they were done with a specific task, they had to tap a "continue" button to go to the next task. In this manner, we can be sure that all the participants followed the exact same activity protocol. The participants were asked to wear an armband with a smartphone on their arm, either a Galaxy S4 or a Nexus 5 running Android 5, so that the smartphone was always with them and it did not interfere with their movement.

A simplified floor plan of the $SmartCondo^{TM}$ space is shown in Figure 3.11. The red stars indicate the locations of the Estimote stickers that were attached to static objects, which cost approximately $10 each. We also attached 12 Estimotes on movable objects used for the script activities, such as a cup, a frying pan, the garbage lid, etc. Moreover, 14 PIR motion sensors, each 3 of them connected to a node (built from scratch in the network's lab at a cost of $20–30 each) were installed on the ceiling, with a Raspberry Pi 3 ($50) nearby to receive the motion sensor events and stream them to the server. The smartphone used costs approximately $150. The phone batteries last approximately 6–7 hours when the accelerometer and magnetometer on the phone are used and the events are streamed to the server.

In keeping with the idea that the sensor-specific localizers can be selected from a wide range of offerings, the actual computational complexity introduced

by our contribution is due to the fusion and post-processing steps. The fusion step involves the addition of the confidence values of different confidence maps produced by different localizers for each individual occupant. This addition takes place over a discretized grid. Hence, if we have "P" individuals, "L" localizers, and "B" grid points in the area, the complexity of the fusion step is $O(P \times L \times B)$. Then, during the post-processing step, the process of confidence reduction for the grid points that are too far from the previous location estimates for each person is performed in $O(P \times B)$. Finally, the disambiguation of anonymous persons involves two phases. First, the confidence maps for each person are clustered together to determine the location-estimate areas ($O(P \times B)$). Next, for each person "p" in the space, for each grid point "b" in the confidence map, the disambiguation method checks if "b" is inside another person's location estimate area, and if so, the confidence of "b" is reduced; this last part can be done in $O(P^2 \times B)$. As a result, the whole process is completed in $O(P \times L \times B + P^2 \times B)$ and since the number of localizers is typically a small constant, decided a-priori and independent of P, the time complexity is essentially $O(P^2 \times B)$. We remark that the generation of each localizer estimate reflected in L can be a significant overhead and varies among localizers.

Extracting the Ground Truth

To collect ground-truth data, the participants' movements and actions were video-recorded by six cameras, also shown in the diagram of Figure 3.11. We subsequently analyzed these videos to annotate them with the ground truth, regarding the participants' activities and locations.

The video annotation was performed manually by myself. I reviewed the videos and recorded the locations of each participant at each point in time, similar to the process outlined in [8]. Although this procedure is bound to produce inaccuracies with respect to the precise timing of each activity and the exact location of the participants, it is currently the only methodological option, given the complexity of the activities and the maturity of current video-analysis methods.

To alleviate the complexity of the ground-truth video-annotation task, the

experiment script given to participants included instructions for them to stand on marked locations on the floor while conducting specific tasks. Not surprisingly, our participants did not follow the instructions very precisely, and for most of the time, they were at unmarked locations for which we do not have exact $[x, y]$ coordinates. During the video-annotation process, we estimated those coordinates based on their location relative to known landmark points around them. Another common source of error in manually establishing the ground truth is introduced when recording the participants' locations while moving: those locations are semi-automatically estimated through interpolation between known timestamped locations.

Another problem with generating the ground truth was the fact that we had three (sometimes four) different sources of timestamps: (a) the smartphones carried by the participants, which send the Estimote events to the server; (b) the database timestamps of the motion-sensor data-transmission events; and (c) the video-recording timestamp. While all those clocks were synchronized at the level of timezone, date, hour, and minute, they were out of sync in terms of seconds. As a result, the timestamps of the ground truth and the inferred location and activity may differ from each other by as much as 60 seconds. To mitigate this problem, we use a time window of length $T = 1, 30, 60$ *seconds* when determining the corresponding ground truth point for each of the system's estimates at each time (see Equation 3.8).

$$Error_{t_i} = min_{t_i-T}^{t_i+T} \| l_{t_i}^{est} - l_{t_i}^{gt} \| \tag{3.8}$$

In the above equation, $l_{t_i}^{est}$ is the estimated location of an occupant at time t_i; $l_{t_i}^{gt}$ is the actual location of the same occupant at the same timestamp, and T is the window's length. Then to calculate the total error and standard deviation for each person, Equation 3.9 is used for all estimates in all the ConfMaps generated during a session for each person.

$$Error_{total} = avg_{t_i}(Error_{t_i}) \tag{3.9}$$

$$STD_{total} = STD_{t_i}(Error_{t_i}) \tag{3.10}$$

There are some computer vision methods, like the one in [33] who try to localize a moving robot in a space with help of a Monte Carlo algorithm. We hope that in the future, we can use these methods to simplify the task of ground truth extraction.

Results and Discussion

In this section, we examine the performance of our Estimote+PIR method. We report and discuss the location-estimate errors, calculated based on the formula in Equation 3.8, for single-participant sessions and two-participant sessions under different knowledge assumptions.

Table 3.2: Localization error for single-participant sessions, using motion-sensor and Bluetooth Low-Energy (BLE)-Estimote data. All the measurements are in meters.

window size	**session 1_1**		**session 1_2**		**session 1_3**	
	mean	std dev	mean	std dev	mean	std dev
1 sec	1.92	1.34	2.35	1.80	2.88	1.72
30 sec	1.52	1.09	1.88	1.53	2.59	1.73
60 sec	1.31	0.86	1.71	1.39	2.50	1.72

Table 3.3: Localization error for single-participant sessions, using motion-sensor data only. All the measurements are in meters.

window size	**session 1_1**		**session 1_2**		**session 1_3**	
	mean	std dev	mean	std dev	mean	std dev
1 sec	2.28	1.35	2.31	1.54	3.28	1.14
30 sec	1.79	1.01	1.77	1.19	3.01	1.45
60 sec	1.58	0.73	1.60	1.05	2.93	1.45

Tables 3.2 through 3.6 show our localization results for six of the 2-hour sessions in our experiment: three of these sessions involved a single participant and the other three involved two participants. The name of the sessions

used in all tables follows the convention "$session1\_i$" to indicate the i-th single participant session. Similarly, "$session2\_i$" is the i -th session with two participants.

Tables 3.2 and 3.3 report the average error of the Estimote+PIR method in three single-participant sessions, under two different conditions: (a) using both motion sensors and Estimotes, and (b) using motion sensors only. Comparing the two tables, one notices the improvement in the localization accuracy that is possible due to the Estimotes. Estimotes improved the localization accuracy by approximately 20 cm on average. This is due to two reasons. First, the union of all the areas covered by the Estimotes is larger than the area covered by the motion sensors (Figure 3.11). Second, and more interestingly, the sensing area of each *individual* Estimote—given our -70 dBm threshold—is relatively smaller than that of the motion sensors, since they are mostly attached to the walls and detect targets within the semi-circle around them. Therefore, when the Estimotes recognize an occupant in their sensing area, they do so with high confidence, and the location estimate becomes more accurate. The standard deviation reported in Table 3.2 is higher than that of Table 3.3. This is because, unlike motion sensors, Estimote errors are not bounded; although we are assuming that RSSI values higher than -70 dBm imply that the target is within one meter of the Estimote, that may not be always the case. According to our previous study in section 3.1 [12], the RSSI value can vary drastically (over a range of 10 dBm or more), even when the target is stationary at a fixed distance; this did not happen frequently in our experiment (so the accuracy is still better when adding Estimotes), but it is sufficient to make the standard deviation slightly higher (28 cm on average when window size was 1). In other words, the coverage area of the Estimote thresholded at -70 dBm is "fuzzy". In contrast, a motion sensor firing means that the individual is within the (bounded) coverage area of the motion sensor.

Tables 3.2 and 3.4 report the average error of Eimote+PIR method in three single-participant sessions and three two-participant sessions, respectively. It is easy to notice that this method exhibits the highest accuracy (and smallest error) when configured with a time-window of 60 seconds. Intuitively, the

coarser the time granularity, the smaller the error. The average localization error when the window size is 60 seconds is somewhere between 0.38 m and 0.64 m (0.5 m on average based on the results from Tables 3.2 through 3.6), better than when a window size of 1 second was used. This fact shows the impact of unsynchronized sensor events on the quality of the method's estimates. Our choice to use a 60-second window is well motivated by the fact that our data-emitting sensors and devices are not synchronized at the granularity of a second.

Table 3.4: Localization error for two-participant sessions, using motion-sensor and BLE-Estimote data, with both participants holding phones. All the measurements are in meters.

window size	**session 2_1**		**session 2_2**		**session 2_3**	
	mean	std dev	mean	std dev	mean	std dev
1 sec	2.42	1.43	2.39	1.70	2.17	1.80
30 sec	2.01	1.19	2.11	1.57	1.82	1.53
60 sec	1.87	1.11	2.00	1.51	1.65	1.37

Table 3.5: Localization error for two-participant sessions, using motion-sensor and BLE-Estimote data with only one participant holding a phone. All the measurements are in meters.

window size	**session 2_1**		**session 2_2**		**session 2_3**	
	mean	std dev	mean	std dev	mean	std dev
1 sec	2.53	1.43	2.49	1.83	2.33	1.77
30 sec	2.06	1.00	2.22	1.73	1.94	1.52
60 sec	1.92	0.90	2.1	1.67	1.77	1.35

Table 3.6: Localization error for two-participant sessions, using BLE-Estimote data only, with both participants holding phones. All the measurements are in meters.

window size	**session 2_1**		**session 2_2**		**session 2_3**	
	mean	std dev	mean	std dev	mean	std dev
1 sec	2.31	1.26	2.38	1.49	1.91	1.36
30 sec	1.92	1.08	2.12	1.38	1.61	1.07
60 sec	1.81	1.06	1.98	1.34	1.46	0.97

For the two-participant sessions, the location-estimation error when both participants wore a smartphone on their arm is reported in Table 3.4. Nevertheless, we are interested in the performance of this method when only some of the participants carry smartphones. This is important for assisted-living facilities, where older adults are unwilling to wear any sensors or carry a smartphone but their caregivers typically have one. To simulate this scenario, we ran two different experiments for each session, ignoring the data emitted by one phone of a participant at a time, and we applied our location-estimation method to the remaining data to examine how effective our method's disambiguation feature is in this reduced-knowledge condition. The average result from the two experiments for each session are reported in Table 3.5. Comparing the results between Tables 3.4 and 3.5, we note a relatively small decline, which provides evidence for the robustness of our method. The participant who does not carry a phone—and as a result is not sensed by the Estimotes—is localized by the motion sensors only, which is possible because the sensing area of each motion sensor is larger than that of the Estimotes: motion sensors sense elements within a diamond around them with the diameter of approximately 2 m, while the Estimotes—due to their on-wall placement—sense within a semicircle of 1 m radius.

For the two-person localization, to the best of our knowledge, all the previous studies required both subjects to wear some kind of sensor or tag, or to carry a smartphone. A 2013 study [38] reported 1.49 m accuracy, but because their method was device-less, it could not disambiguate the occupants. Mak-

ing even more stringent hardware assumptions, a 2009 study [15] reported an error of 1 cm only, but required RF transmitters and receivers and assumed sensors wired to the transmitters carried by the occupants. In this scenario, the batteries of sensors mounted on different body parts lasted only about 1–2 hours, and the coverage area of each transmitter was only 3 m and was sensitive to the presence of metal objects in the area. Clearly, even though the obtained error is quite impressive, the method cannot be applied in any real-world scenario.

In our experiment, we were able to achieve almost the same accuracy when only one of the two participants carried a smartphone (Tables 3.4 and 3.5). When space was equipped with motion sensors, our Estimote+PIR method was able to still infer the likely locations of the two participants and relies on the single source of eponymous data to disambiguate those locations.

Table 3.6 presents the errors obtained for the same two-participant sessions when only Estimotes were used, without taking any motion sensor data into account. Remarkably, the location estimate errors are better than those reported in Table 3.4 by approximately 10 cm. This is due to the larger area coverage resulting from the union of the individual covered areas by the deployed Estimotes—approximately 40 m^2—compared to the 26 m^2 collectively covered by the motion sensors.

In a simple mutation experiment, we eliminated every other Estimote sensor and recomputed the location-estimate error: the average error for both participants for session 2_3 became 3.24 m (for a time window of 60 seconds), which is worse than the result reported for the same session in Table 3.4 or 3.6. This confirms our intuition that the superior accuracy of the Estimotes-only location estimates is an artifact of their deployment density.

Besides evaluating the accuracy of the location estimates produced by our Estimote+PIR method, we also analyzed its confidence. As we explained above, the output of the localization process is a confidence map for each person at each point in time, such as the one shown in Figure 3.10. Based on this confidence map, we also computed an overall confidence measure for the estimates produced by our method. As we discussed before, the confidence

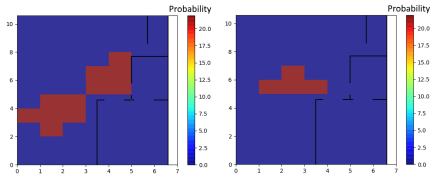
maps assign a value to each point in the monitored space, shown by colours in Figure 3.10: the deeper the colour, the more confident our method is that there is someone—whether anonymous or eponymous—in that point. The overall confidence measure we discuss here is the maximum of these values over the whole map. Our hypothesis is that the high errors (more than 4 m) are due to prolonged periods without data. Indeed, there were time periods throughout the sessions during which the server did not receive events from our sensors despite the participants' movements. The facility where the experiments were conducted is awash with RF interference, and possible network throughput deterioration and even outages are within the realm of possibility. Indeed, our hypothesis on the origins of the inaccuracy and low overall confidence is validated in Figure 3.12. This Figure demonstrates that the method's confidence is very low when the error is high, which implies that our method is "aware" of its blind spots. More precisely, when our Estimote+PIR method lacks input from sensors, based on Algorithm 1, the most recent confidence map is used but referring to Algorithm 4, the system reduces the confidence values. Hence, if the participant has been moving to a new location during the period for which the server did not receive any data-transmission events, the confidence would be low. In Figure 3.12 you can see this effect where in case of higher error for a long period of time (shown in the red-dotted areas in Figure 3.12a), the corresponding confidence measure is low (shown in the red-dotted areas in Figure 3.12b).

Finally, we conducted a preliminary analysis of our Estimote+PIR method's effectiveness for activity recognition. There is a fairly limited set of activities that we are able to detect in our data. By attaching Estimotes on the objects shown in the left column of Table 3.7, we are able to recognize basic activities relying on the person's interaction with the object in question. For example, when the Estimote attached to the iron is recognized by a participant's smartphone, our method infers that the person is ironing. Several basic activities are grouped under a single "activity" header. For example, using an iron or a laundry basket or a washer or a dryer implies that the person is "doing laundry". During all of the abovementioned sessions, our method was able to

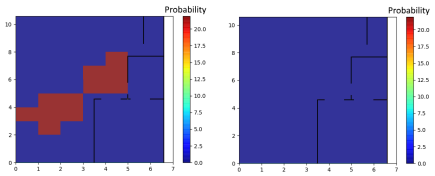
correctly detect 70% of the activities that the occupants were doing.

Table 3.7: Activities recognized based on Estimotes attached to objects.

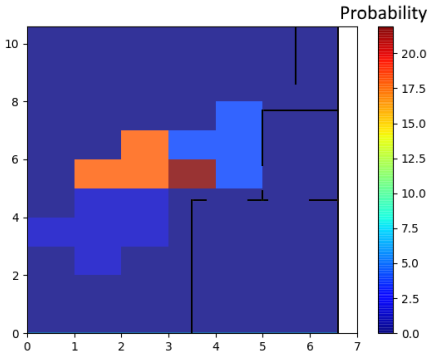
Basic Activities	**Activities**
Use iron, Use ironing board, Move laundry basket, Use dryer	Laundry
Use dustpan, Use broom	Brooming
Use TV remote	Watching TV
Use kettle, Use frying pan, Use cup, Open/Close kitchen cabinet, Open/Close fridge, Open/Close garbage lid	Cooking/Eating/Washing Dishes
Take medication	Medication



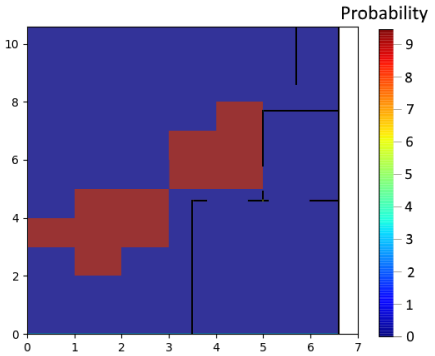
(a) Initial confidence maps produced by motion localizer (right) and estimote localizer (left) for participant 1



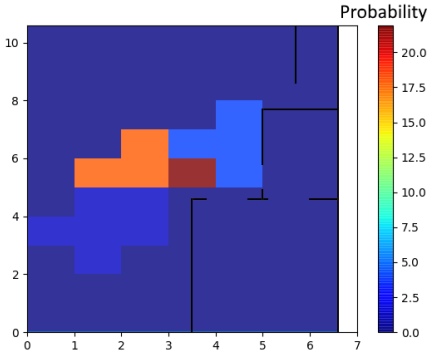
(b) Initial confidence maps produced by motion localizer (right) and estimote localizer (left) for participant 2



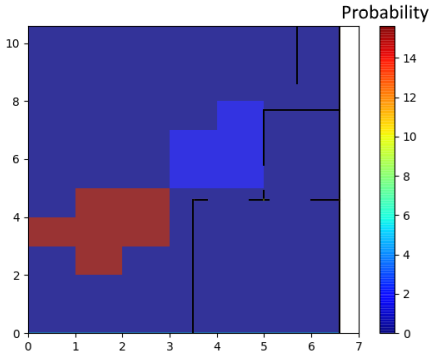
(c) Confidence map for participant 1 after fusion



(d) Confidence map for participant 2 after fusion



(e) Final confidence map for participant 1



(f) Final confidence map for participant 2

Figure 3.10: Confidence maps for 2 persons with motion sensors, and Estimote events for person 1 only. Figure 3.10a, 3.10c, and 3.10e correspond to participant 1, and Figures 3.10b, 3.10d, and 3.10f correspond to participant 2.

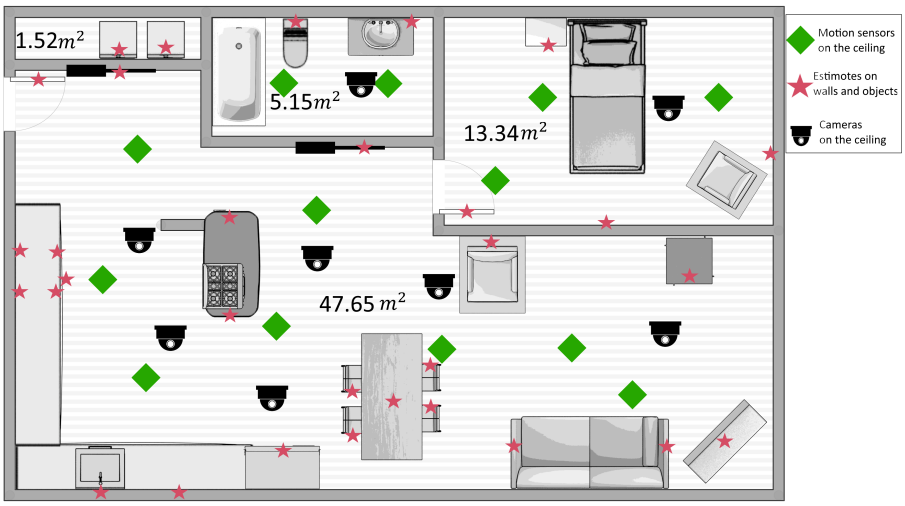
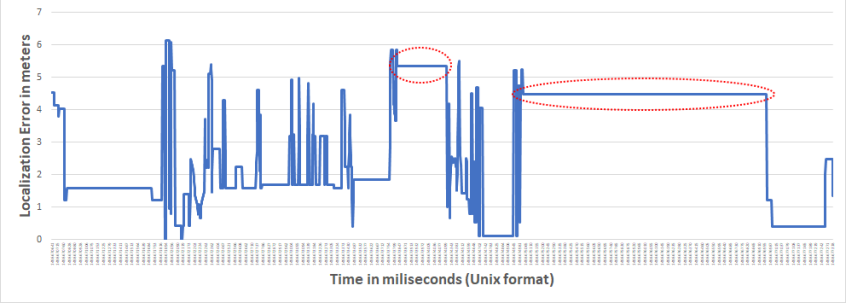
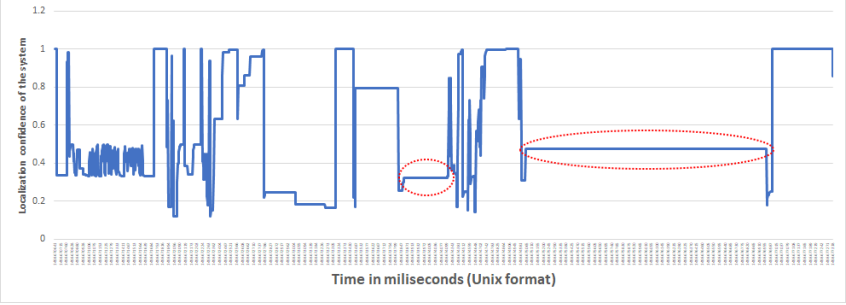


Figure 3.11: Layout of the *SmartCondo*^{TM} with positions of static Estimote stickers and PIR motion sensors.



(a) Error of the localization for one of the single occupant sessions.



(b) Confidence of the system on localizing one of the single occupant sessions.

Figure 3.12: (Figures **a**) and (**b**) are from the same single occupant session.

Chapter 4

The $SmartCondo^{TM}$ Software Architecture

The design we mentioned in section 3.2 was used at the development phase and when the experiments have been done. After testing the system and verifying its effectiveness, we evolved its software architecture so that it meets the following requirements

- **Real-time location estimation:** It needs to localize the occupant, based on the events receives from sensors, in a real-time manner.
- **Scalability:** There are three types of scalability: size scalability, geographical scalability, and administrative scalability. We will describe how our system provides any of these types of scalability later in this chapter.
- **Configurability:** There are many parameters in the methodology we used for indoor localization that should be easy to alter in the future.
- **Easy of Deployment:** Because of the nature of the system, it needs to be re-deployed in different buildings with different sensors.

This chapter explains how we changed our system described in section 3.2 to have the above characteristics. Figure 4.1 shows the architecture of the final system. With this design, the Algorithm 2 and Algorithm 3 reside in their own separate web services, and the Algorithm 4 and the post-processing step

are in smart condo web service. Finally, the Algorithm 1 is divided into two parts separated into the two localizer web services.

For the system to be real-time, we observed that it takes an average of 4 seconds to localize 2 persons in the environment at each timestamp. As we have our sensors sending events approximately every 1 second, we decided to make the system run the localization every 3 seconds so that we will not lose as much data and the system will be acting almost real-time.

For scalability, our system provides geographic scalability when it comes to adding new types of sensors. Each type of sensor will send its events to the corresponding web services through a RESTfull API. By separating the localizers, in the future, one can create a new localizer for a new set of sensors that receives events from its dedicated sensors and then perform its own algorithm. This is good in many aspects. First, new localizers do not need to be written in the programming language that the current system is implemented with (python), and doesn't even need to be on the same server computer as any of the current localizers. Second, we believe that different sensors have different characteristics and the information we can receive from them is not the same. With the current design of the software, the functionality of the new sensor's localizer can be completely different from the other localizers.

The only thing that matters is that the output of all localizers should be of the same format. This output will be sent to the smart condo web service somewhere else on the cloud. The communication of localizers with the smart condo web service is through another RESTfull API which will be discussed later in this chapter. This API also is provided to any visualization system which wants to request location estimates from the system and visualize it. All these works has made the software very scalable in terms of adding new sensors and new visualizer systems.

The size scalability of our system is related to the number of occupants increases. According to the time complexity calculated in section 3.2, our software will be polynomially scalable when we add new occupants. Finally, the administrative scalability of our system depends on its ability to be deployed in multiple locations. Right now, the preferred way to do that is by

installing multiple systems, one per each new space. This is not the optimal way of scaling our system and we believe that in the future, we can modify some small parts of the design in order to add multiple spaces in the same system.

For the system to be configurable, every parameter in the code is moved to configuration files. These parameters include the size of the grid we use for localizing a person in a building, the step size of the A^* algorithm, etc.

For the system to be easy to deploy, all the information about the map of the space, the sensors installed in the area, and the URL's for the smart condo web service, where the localizers should connect to, are in the configuration files that are in localizers and smart condo web services. The devices that should communicate with the localizers (Raspberry Pi and Android phones) will request for the configuration files once they want to start working with the system and will receive the information they need about the sensors that they should be listening to.

The smart condo web service is also responsible for saving all the raw events and the output location and activity estimates from different localizers and the fused confidence maps. The design of the DataBase is described in the next section.

4.1 DataBase Design

We decided to use a relational (or SQL) database (Postgres) mainly because of three reasons. First, we implemented the scheme of the database to be general for different technologies. For example, the table of sensors is defined to be able to contain various types of sensors so that adding new sensors in our system means adding a new row to the sensor table and not a new table. As a result, our system only needs to be scalable vertically which is suitable for SQL databases. Second, the queries that we needed to use in our system are handled much faster in SQL databases; and third, because of the high load of data being streamed to our server, an SQL database was a better choice for our intended applications.

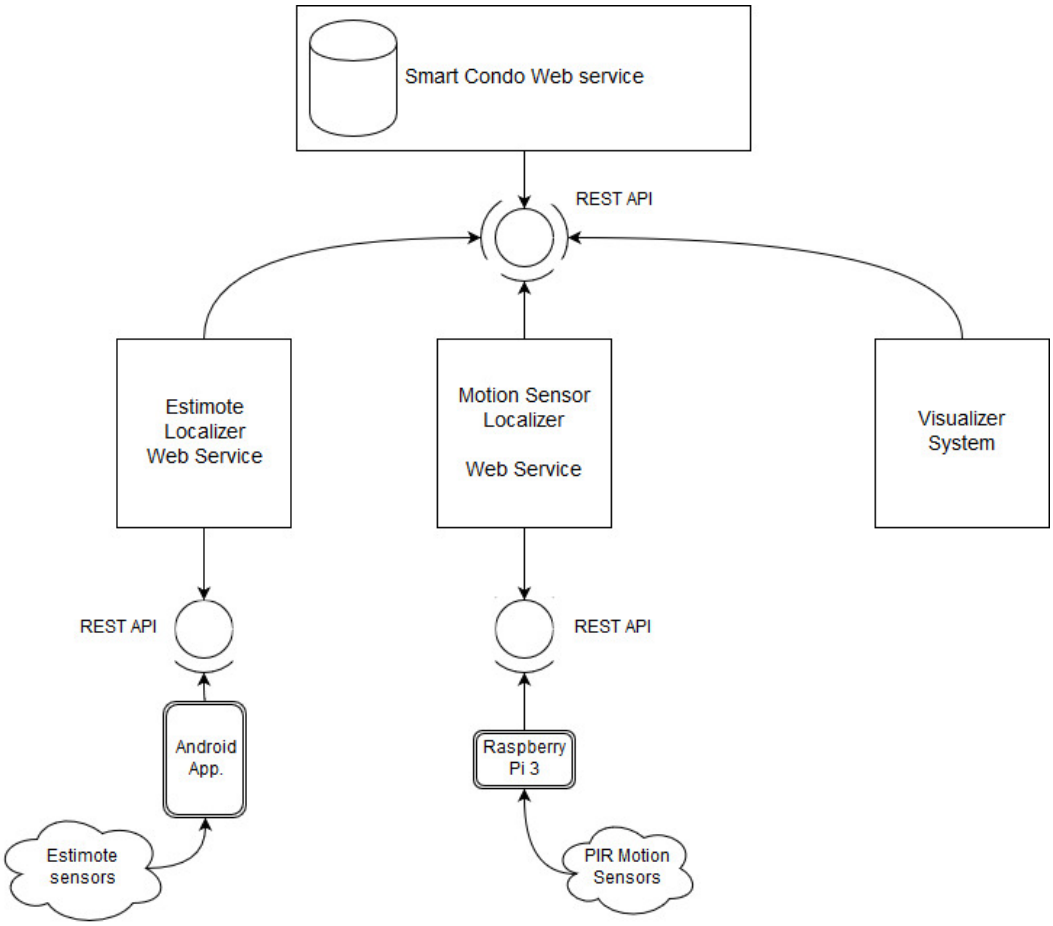


Figure 4.1: Final design of the $SmartCondo^{TM}$system

The list of tables of our database are as following:

- **Sensor:** This table has the information about different sensors in the environment. Each row of the table has the following fields:
  - **hash:** This is a unique ID for each sensor.
  - **sensor type:** The type of sensors like "Estimote" or "Motion" in our case.
  - **x, y, z:** The location of the sensor in the environment. This values can be empty when a sensor is used only for activity recognition and does not have a fix location, e.g. an Estimote attached to a book.
  - **attached object:** The name of the object that the sensor is at-

tached to.

  - **sensing area:** The corners of the polygon that the sensor can sense when there is a person in there.
  - **description:** A text including more useful information about the sensor.

- **Person:** This table has the information about each person in the environment who is registered to be localized by our system. Each row of the table has the following fields:
  - **hash:** A unique ID for each person.
  - **name:** The name of the person.
  - **device ID:** The ID of the device that this specific person uses.
  - **speed:** The speed of the person when he/she is walking in the area.

- **Event:** This table has the information about each event that we receive from any sensor. The format of this table is designed to be general for every sensor event possible. Each row of the table has the following fields:
  - **hash:** A unique ID for each event.
  - **source:** Where the event comes from, e.g. "Estimote" or "Motion" in our study.
  - **timestamp:** The time of the event in Unix time format.
  - **sensor:** This is a foreign key to one of the rows in Sensor table, indicating which sensor the event is coming from.
  - **person:** This is a foreign key to one item of the Person table, indicating for which person this event is. This value can be empty too because some of the sensors, e.g. motion sensors, cannot identify the person within their sensing area.
  - **data:** For the purpose of making the Event table general for all possible sensors, the information we receive from each sensor is in

this field in a JSON format. For example in our case, the data field of the events coming from Estimote sensors has information about RSSI, Accelerometer, temperature, etc.

- **Confidence Map:** As mentioned before, the output of each localizer and the whole system is a confidence map for each person at each timestamp. Each row of the table has the following fields:
  - **hash:** A unique ID for each event.
  - **source:** Where the confidence map comes from, e.g. "Estimote" or "Motion" or "fusion" or "post-process" in our study.
  - **timestamp:** The time of the confidence map in Unix time format.
  - **person:** This is a foreign key to an item of the Person table, indicating for which person this confidence map is.
  - **map:** A JSON format data which has an array of grids locations and the confidence value for the specific person being there.
  - **source reliability:** The reliability of the localizer which is used when fusing confidence maps together.
  - **estimate confidence:** The confidence of the generated estimates.
  - **x, y:** The location of the person.
- **Activity:** Each row of the table has the following fields:
  - **hash:** A unique ID for each event.
  - **source:** Where the activity inference comes from, e.g. "Estimote" in our study.
  - **timestamp:** The time of the activity in Unix time format.
  - **person:** This is a foreign key to an item of the Person table, indicating for which person is this activity.
  - **Activity:** A text indicating what the person is doing.

4.2 Web API Design

According to the Figure 4.1, the system has 3 APIs, one for the localizers to communicate with the smart condo web service and one for the Android application to communicate with the Estimote localizer and one for the Raspberry Pi to communicate with the Motion localizer. This section describes these 3 APIs.

The smart-condo service exposes an API to the localizers, with the following operations.

- **Add event:** The localizers will send the raw events coming from sensors to the smart condo service to be saved in the database. This operation receives a timestamp and the data of the event in the form of JSON.

- **Get config:** The localizers get the configuration files from the server with this operation. It receives a sensor type from the localizer (e.g., Estimote or Motion in our case) and in the results, sends sensors' information, the map of the space, and the network configuration data which tells the localizer differents URL addresses for the API.

- **Add person:** The Estimote localizer sends information of the new person registered by his/her smartphone to the smart condo. This information includes device ID and the username. In return, the server sends the person hash ID to the phone which will be used in the future communications.

- **Add confidence map:** Different localizers send their output confidence maps in the format of a JSON to the server. This JSON value has the information about the timestamp, person, x and y location, the confidence of the estimate, activity, and the confidence map. Any of these values except timestamp can be empty. For example, the localizer may have only the activity information of an occupant, or in case of the motion sensor localizer, the person can be empty.

- **Add stats:** The Raspberry Pi sends its state every 10 minutes to the smart condo service containing information about its battery level, disk and memory space.

The Estimote localizer provides an API to the Android application and the Motion localizer also provides an API to the Raspberry Pi. The APIs are almost identical, supporting two operations.

- **Add event:** The Raspberry Pi/Android application sends motion sensor/Estimote sensor events to the localizer. The parameters are a timestamp and a data in the format of a JSON string containing the information about the event.

- **Get config:** For the Motion localizer, the Raspberry Pi requests the configuration file including sensors information and the network URLs information from the localizer that is returned in a JSON format.
  For the Estimote localizer, the Android application sends its device ID and the user's name to the localizer to register the person and in return, receives the configuration file including the map coordinates, the Estimote sensors information, and the network URLs information.

When a new sensor type needs to be added to the system, a new localizer web service needs to be deployed for it. The web service should be deployed somewhere on the cloud and communicate to the smart condo web service with the API described in this section. The way the new localizer communicates with its sensors is completely up to the person implementing it. It can have the same API as the current Estimote and Motion localizers but also can have a completely different format to best suit the new sensors and their characteristics. The only important rule that has to be followed is the way it communicates to the smart condo web service and the format of the data that is being sent to it.

4.3 Deployment Process

As mentioned before in this chapter, the smart condo software is easy to deploy in new environments with new sensors. The deployment process is done in 5 steps:

1. **Install sensors:** At first, the sensors need to be purchased and installed preferably in a way that they cover all the space. Estimote sensors can be purchased from their website [19]. The PIR motion sensors were built from scratch in the network lab at the University of Alberta.

2. **Prepare the configuration files:** Configuration files are in text format and reside on the same servers as the smart condo and localizers web services are deployed. We call them smart condo server and localizer servers. At the smart condo server, there are 3 configuration files that need to be changed properly:

    - **network configuration.txt:** This file contains the information about smart condo API URLs, the size of the grid that should be used for localization, the interval that the Raspberry Pi should send its statistics, and a list of localizers communicating to the smart condo web service (e.g., ['ESTIMOTE','MOTION']). The name of the localizers in the later field should match the 'source' field that the localizer is using when making a confidence map (refer to confidence map in section 4.1).

    - **map.txt:** This file has the coordinates of the map of the space and the obstacles. The obstacles can be defined as polygons with their corners' coordinates.

    - **estimote.txt:** This file contains the information about all Estimote sensors that are deployed in the space.

    - **motion.txt** This file contains the information about all PIR motion sensors that are deployed in the space.

When a new type of sensor is added to the system, there should be a new configuration file in the smart condo server containing the information about all those sensors which are deployed in the area.

At each of the localizers, there is another set of configuration files:

- **network configuration.txt:** This file includes the information about the specific localizer's API URLs.

3. **Set up the servers:** One can deploy all the web services on the same server listening on different ports, or on different machines. To deploy them, an Ubuntu 14 or higher is needed. The setup is very straightforward and a setup guide is prepared in order to do that.

4. **Set up the Raspberry Pi:** A python program, developed in the network lab, should be deployed on the Raspberry Pi and the IP address of the server on which the motion localizer is deployed should be set in its code.

5. **Install the Android application:** As mentioned before, the Estimote sensors talk to an Android application. This application is developed using the Estimote SDK provided by the company which uses the Bluetooth Low Energy technology to get the packets from Estimotes. Based on some research on the Estimote online forum, we realized that some of the older Android phones do not work properly with the Estimotes. The phone needs to have Android version of 5 or higher and should not be in the list of incompatible devices that we prepared based on trial and error by ourselves or other users. After installing the application, the user should turn on WiFi and Bluetooth. The user is then asked to enter a username and the URL of the server that the Estimote localizer is deployed on, in order to register the device. Then the application loads the map of the space and while he/she is moving around in the sensing area of an Estimote sensor, the new sensor is shown to the user as well.

Chapter 5

Conclusion

5.1 Conclusion and Future Work

In this thesis, we have addressed the problem of estimating the location of multiple individuals moving and interacting in an indoor space. Our work makes three key contributions. First, it proposes *a multi-sensor data-fusion framework*, relying on a unifying location-estimate representation as a *confidence map* of the indoor space. In this framework, each distinct type of sensor data is processed by a sensor-specific algorithm to generate a sensor-specific *confidence map*; all sensor-specific confidence maps are subsequently fused into a single set of confidence maps corresponding to location estimates for each individual. Second, *our framework distinguishes between anonymous and eponymous sensors*, such as motion sensors and Estimote stickers. This combination enables our Estimote+PIR method to accurately recognize individuals when all, or in 2 participant session, only one, carry a smartphone running the application that collects the Estimote sensor events. The later scenario is extremely important because it is motivated by the requirements of real settings involving caregivers who are willing to adopt and carry technologies such as smartphones, but the cared-for person is unable or unwilling to consistently use such a device. Third, *our framework is implemented in an extendible and easy-to-deploy software system*, enabling it to be forward compatible with new sensors, as long as they can be categorized in the two categories above. We believe that in the future more accurate sensors, especially for the purpose of activity recognition, can be added to the framework, and by using the location

estimates coming from other sources, it can detect the activity more precisely.

We conducted six experiments, involving data collection from a real environment. We established that using our Estimote+PIR method, the location of the individual that is not carrying/wearing a device on them can be determined just as accurately as it would have been if the individual was carrying a device. We also identified several crucial parameters that influence the accuracy of the proposed scheme. One of them is the relative coverage of the space by the sensing "footprint" of the eponymous data collected via the Estimotes: the larger this space is, and the higher the number of Estimotes deployed, the smaller is the impact of the coverage by the anonymous (motion) sensors. Nevertheless, this has to be seen against the backdrop of deciding on an RSSI threshold for the Estimote signals (-70 dBm in this study), which effectively transforms the Estimotes into proximity sensors. We also noticed that the lack of synchronization across the two (and potentially more) sources of sensed data, if not addressed at a lower layer, has to be accommodated when defining the accuracy metrics of a "fused" location-estimation scheme. By affording a 60-second window delay, we could derive more accurate location estimations than for shorter delays. The situation may be quite typical in future systems that use completely heterogeneous sources of data, utilizing different technologies and communication standards. In most such cases, no single one-size-fits-all low-level synchronization solution can be used, and the onus of synchronization shifts to the application layer.

We have conducted and reported on a number of experiments demonstrating the efficacy of our Estimote+PIR method, but we also note that a significant cost of the overall endeavour was the ground-truth annotation of the source data. To the extent that one high-priority item that impedes future research can be described, it will have to be the ability to automatically (or at least semi-automatically) annotate ground truth from captured traces. The use of computer vision techniques may be indispensable for such a task.

Finally, by re-designing the software, we were able to do the localization in real-time. Moreover, the software is now much easier to scale and add new technologies.

Tables A.1 and A.2 in appendix A compares our Estimote+PIR method and recent related work in this area. The Tables demonstrates that our method makes realistic knowledge assumptions, does not require onerous configuration-deployment effort, and has been thoroughly evaluated in realistic scenarios. Almost all the methods that have better accuracy than ours have not been tested in real environment when the subject has natural movements of an everyday life. Our system, as we discussed in chapter 3.2.6, have been evaluated in the $SmartCondo^{TM}$ with single or double subjects doing everyday activities.

5.2 Future Work

An interesting subject for the future would be using the history of the location and activities of the occupants into account when making an inference on the current location (e.g., it is more likely to go to the dining room after making tea than to the bathroom). Right now, we are using history in a very simple way with only two purposes, first, comparing current location with the last one, the person cannot move faster than 1.3 m/s. Second, when there is no new event coming from one of the groups of sensors, the system fuses the last data available with the new data coming from the rest of the sensors.

Another interesting area which needs more attention in the future is the threshold that we have used for the Estimote sensors. Based on our experiments in section 3.1, we decided to use -70 dBm threshold for when the subject is in 1-meter distance from the sensor. We believe that this needs more research and by doing a profiling strategy on every new space that the system is being deployed, and also taking into account the fact that the RSSI distribution can change based on the power level of the transmitter [28], we can select the optimum value for threshold and as a result of that, the accuracy of the localization system can improve.

In this research, we didn't put much effort on the activity recognition task. We only used the Estimote sensors for that and only took the accelerometer values to detect their movement. In the future, other than using other types of sensors like switches and pressures sensors, and using other properties of

the Estimote data, we can use more complex methods like machine learning algorithms to do the activity recognition like the ones used in [10]. These methods can use the location of the occupants as an input feature to better guess the activity, e.g., it is not possible for a person in the bedroom to be using the stove in the kitchen.

Finally, based on our observations during this study, we believe that the battery lifetime can be improved further by sending a "stop" message from the server to the phone, to stop scanning for and streaming Estimote data when the caregiver's phone is far away from the area where the cared-for person may be located. This condition can be identified when the Google geolocation API localizes the device away from the residence of the patient and no relevant Estimote readings have been received for some time. Another frequently employed technique is to identify from accelerometer data that the phone or smartphone is stationary and to throttle or stop sending updates, but such inactivity might mean that the individual has forgotten or not worn the device, and the protocol to react to such exceptions is dependent on the exact context. This feature—namely, controlling the phone application operation for the sake of extending its battery life—can be the subject of the possible future work.

References

[1] F. Adib, Z. Kabelac, and D. Katabi, "Multi-person localization via rf body reflections.," in *NSDI*, 2015, pp. 279–292. 8

[2] M.-H. Amri, Y. Becis, D. Aubry, N. Ramdani, and M. Fränzle, "Robust indoor location tracking of multiple inhabitants using only binary sensors," in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, IEEE, 2015, pp. 194–199. 10, 69

[3] D. Ayllón, H. A. Sánchez-Hevia, R. Gil-Pita, M. U. Manso, and M. R. Zurera, "Indoor blind localization of smartphones by means of sensor data fusion," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 4, pp. 783–794, 2016. 8, 69

[4] M. V. Azghandi, I. Nikolaidis, and E. Stroulia, "Sensor placement for indoor multi-occupant tracking," in *Information, Intelligence, Systems and Applications (IISA), 2015 6th International Conference on*, IEEE, 2015, pp. 1–8. 2, 3, 12, 35, 68

[5] S. Beauregard and H. Haas, "Pedestrian dead reckoning: A basis for personal positioning," in *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication*, 2006, pp. 27–35. 6

[6] Z. Chen, Q. Zhu, and Y. C. Soh, "Smartphone inertial sensor-based indoor localization and tracking with ibeacon corrections," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 4, pp. 1540–1549, 2016. 7, 68

[7] T. Chowdhury, M. Rahman, S.-A. Parvez, A. Alam, A. Basher, A. Alam, and S. Rizwan, "A multi-step approach for rssi-based distance estimation using smartphones," in *Networking Systems and Security (NSysS), 2015 International Conference on*, IEEE, 2015, pp. 1–5. 5, 15

[8] P. N. Dawadi, D. J. Cook, and M. Schmitter-Edgecombe, "Automated cognitive health assessment from smart home-based behavior data," *IEEE journal of biomedical and health informatics*, vol. 20, no. 4, pp. 1188–1194, 2016. 38

[9] S. Depatla, A. Muralidharan, and Y. Mostofi, "Occupancy estimation using only wifi power measurements," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 7, pp. 1381–1393, 2015. 8

[10] D. Diaz, N. Yee, C. Daum, E. Stroulia, and L. Liu, "Activity classification in independent living environment with jins meme eyewear," in *Percomm 2018*, Not yet published. 62

[11] M. P. Fanti, G. Faraut, J.-J. Lesage, and M. Roccotelli, "An integrated framework for binary sensor placement and inhabitants location tracking," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016. 10

[12] S. Ferdous, E. Becker, L. Fegaras, and F. Makedon, "Multi-person identification and localization using rfid and passive sensor technology," in *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, ACM, 2011, p. 66. 10, 69

[13] C. Fraley and A. E. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," *The computer journal*, vol. 41, no. 8, pp. 578–588, 1998. 36

[14] G. Galatas, S. Ferdous, and F. Makedon, "Multi-person identification and localization for ambient assistive living.," *LECTURE NOTES IN COMPUTER SCIENCE*, no. 8028, pp. 109–114, 2013, ISSN: 03029743. 10, 68

[15] Y. Gu, A. Lo, and I. Niemegeers, "A survey of indoor positioning systems for wireless personal networks," *IEEE Communications surveys & tutorials*, vol. 11, no. 1, pp. 13–32, 2009. 44

[16] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. 34

[17] F. Höflinger, R. Zhang, J. Hoppe, A. Bannoura, L. M. Reindl, J. Wendeberg, M. Bührer, and C. Schindelhauer, "Acoustic self-calibrating system for indoor smartphone tracking (assist)," in *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, IEEE, 2012, pp. 1–9. 9, 69

[18] C.-C. Hsu and J.-H. Chen, "A novel sensor-assisted rfid-based indoor tracking system for the elderly living alone," *Sensors*, vol. 11, no. 12, pp. 10 094–10 113, Oct. 2011, ISSN: 1424-8220. DOI: 10.3390/s111110094. 7, 69

[19] E. Inc. (). Estimote website, [Online]. Available: www.estimote.com. 57

[20] S. Kumar and R. M. Hegde, "Multi-sensor data fusion methods for indoor localization under collinear ambiguity," *Pervasive and Mobile Computing*, vol. 30, pp. 18–31, 2016. 9, 69

[21] E.-E.-L. Lau, B.-G. Lee, S.-C. Lee, and W.-Y. Chung, "Enhanced rssi-based high accuracy real-time user location tracking system for indoor and outdoor environments," 2008. 14

[22] X.-Y. Lin, T.-W. Ho, C.-C. Fang, Z.-S. Yen, B.-J. Yang, and F. Lai, "A mobile indoor positioning system based on ibeacon technology," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2015, pp. 4970–4973. 6

[23] E. S. Lohan, J. Talvitie, P. F. e Silva, H. Nurminen, S. Ali-Löytty, and R. Piché, "Received signal strength models for wlan and ble-based indoor positioning in multi-floor buildings," in *2015 International Conference on Location and GNSS (ICL-GNSS)*, IEEE, 2015, pp. 1–6. 5, 6, 15

[24] P. Mohebbi, E. Stroulia, and I. Nikolaidis, "Indoor localization: A cost-effectiveness vs. accuracy study," in *Computer-Based Medical Systems (CBMS), 2017 IEEE 30th International Symposium on*, IEEE, 2017, pp. 552–557. 8, 32, 68

[25] ——, "Sensor-data fusion for multi-person indoor location estimation," *Sensors*, vol. 17(10), no. 2377, 2017, ISSN: 1424-8220. 4

[26] N. Roy, A. Misra, and D. Cook, "Ambient and smartphone sensor assisted adl recognition in multi-inhabitant smart environments," *Journal of ambient intelligence and humanized computing*, vol. 7, no. 1, pp. 1–19, 2016. 3

[27] W. Ruan, Q. Z. Sheng, L. Yao, L. Yang, and T. Gu, "Hoi-loc: Towards unobstructive human localization with probabilistic multi-sensor fusion," in *Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on*, IEEE, 2016, pp. 1–4. 6, 68

[28] V. Shrivastava, D. Agrawal, A. Mishra, S. Banerjee, and T. Nadeem, "Understanding the limitations of transmit power control for indoor wlans," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ACM, 2007, pp. 351–364. 61

[29] M. Soleimanifar, X. Shen, M. Lu, and I. Nikolaidis, "Applying received signal strength based methods for indoor positioning and tracking in construction applications," *Canadian Journal of Civil Engineering*, vol. 41, no. 8, pp. 703–716, 2014. 6, 16

[30] M. Srbinovska, C. Gavrovski, and V. Dimcev, "Localization estimation system using measurement of rssi based on zigbee standard," in *Conference Proceedings of the 17th International Scientific and Applied Science Conference (Electronics 2008)*, 2008, pp. 48–50. 16

[31] S. Stieber, R. Dorsch, and C. Haubelt, "Accurate sample time reconstruction for sensor data synchronization," in *International Conference on Architecture of Computing Systems*, Springer, 2016, pp. 185–196. 9

[32] J. Talvitie, M. Renfors, and E. S. Lohan, "Distance-based interpolation and extrapolation methods for rss-based localization with indoor wireless signals," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 4, pp. 1340–1353, 2015. 5

[33] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001. 40

[34] J. Torres-Sospedra, A. R. Jiménez, S. Knauth, A. Moreira, Y. Beer, T. Fetzer, V.-C. Ta, R. Montoliu, F. Seco, G. M. Mendoza-Silva, *et al.*, "The smartphone-based offline indoor location competition at ipin 2016: Analysis and future work," *Sensors*, vol. 17, no. 3, p. 557, 2017. 7, 68

[35] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-level localization with a single wifi access point.," in *NSDI*, 2016, pp. 165–178. 8

[36] I. Vlasenko, I. Nikolaidis, and E. Stroulia, "The smart-condo: Optimizing sensor placement for indoor localization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 436–453, 2015. 2, 11, 12, 32, 69

[37] J. Wendeberg, F. Hollinger, C. Schindelhauer, and L. Reindl, "Anchor-free tdoa self-localization," in *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*, IEEE, 2011, pp. 1–10. 9

[38] C. Xu, B. Firner, R. S. Moore, Y. Zhang, W. Trappe, R. Howard, F. Zhang, and N. An, "Scpl: Indoor device-free multi-subject counting and localization using radio signal strength," in *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*, IEEE, 2013, pp. 79–90. 3, 43, 68

[39] X. Xu, M. Wang, L. Luo, Z. Meng, and E. Wang, "An indoor pedestrian localization algorithm based on multi-sensor information fusion," *Journal of Computer and Communications*, vol. 5, pp. 102–115, 2017. 7, 69

[40] H. Zou, Z. Chen, H. Jiang, L. Xie, and C. Spanos, "Accurate indoor localization and tracking using mobile phone inertial sensors, wifi and ibeacon," in *Inertial Sensors and Systems (INERTIAL), 2017 IEEE International Symposium on*, IEEE, 2017, pp. 1–4. 7, 68

Appendix A

Method Comparison

Tables A.1 and A.2 are a summary of all the previous methods discussed in chapter 2 and our Estimote+PIR method.

Table A.1: Comparing Indoor Localization Methods

	Our Estimote+PIR Method	**ruan[27]**	**chen[6]**	**torres[34][1]**	**zou[40]**	**xu[38]**	**galatas[14]**	**mohebbi[24]**	**azghandi[4]**
uses wearables	yes	yes	yes	yes	yes	no	yes	yes	yes
needs sensor location	yes	yes	yes	yes	yes	yes	yes	yes	yes
requires training, fingerprinting, model fitting	no	yes	yes	yes	yes	yes	no	yes	no
number of test scenarios with people	6	3	2	1	1	1	2	3	0
length of each experiment	120 mins	1 mins	NR	NR	NR	NR	NR	30 60 mins	NR
natural movements during experiment	yes	yes	no	no	no	no	yes	yes	simulation
mean error of single person localization	1.8[2]	0.58	1.28-1.39	2	0.59	1.3	NR [3]	1.8	NR
mean error of multi-person localization	1.8[4]	NR	NR	NR	NR	NR	67-87% detection in room	NR	0.6 - 1.6
tachnology	PIR Motion, Estimote smartphone	RFID (tags, anthenas, reader), switches, lights	smartphone, WiFi, iBeacon	smartphone, WiFi	smartphone, WiFi, iBeacon	Radio transmitter and receiver	MS Kinect, Alien ALR-9900, RFID tag and reader, Alien ALR-9611 circular polarization antenna	smartphone, Estimote, WiFi	PIR Motion, RFID tags and reader
algorithm	Fuse data from sensors and generate a confidence map over space for both participants at each time	fingerprint, sensor fusion, probability matrics of transition states	PDR, WPL(weighted path loss) for initial positions	PDR, wifi fingerprinting	PDR, path loss model for iBeacon RSSI, fingerprinting for wifi	fingerprinting for radio signals	skeletal tracking with kinect, uses RFID tags to disambiguate	trilateration of RSSI values converted to distance by a path loss model	use motion sensors to make the trajectories of people, then uses RFID events to disambiguate tracks

[1]Best method at the IPIN 2016, HFTS Team
[2]window size = 60sec
[3]Not Reported
[4]window size = 60sec

Table A.2: Comparing Indoor Localization Methods. Cont.

	Xiangyu[39]	**Kumar [20]**	**Ayllon [3]**	**Hoflinger [17]**	**Amri [2]**	**Hsu [18]**	**Ferdous [12]**	**Clasenko [36]**
uses wearables	yes	yes	yes	yes	no	yes	yes	no
needs sensor location	yes	yes	no	no	yes	yes	yes	yes
requires training, fingerprinting, model fitting	yes	yes	no	no	no	yes	no	no
number of test scenarios with people	1	0	0	1	1	1	0	0
length of each experiment	NR	static devices	static devices	1 person move on a 14m path	NR (2 person in a home)	8 sec	NR	simulation
natural movements during experiment	no	no	no	no	yes	no	no	yes
mean error of single person localization	0.27-0.35	1.9	0.1-0.3	0.3	NR	0.5	NR	0.4-1.4
mean error of multi-person localization	NR	1.9	0.1-0.3	NR	Just counting	NR	NR	NR
tachnology	smartphone WiFi	Crossbow motes MTS310, XM2110 IRIS board, MIB520 USB mote interface, sound, light sources"	smartphone	smartphone sound receiver	PIR motion	RFID tags and readers, three-axis wireless acceleromer	RFID tags and readers	PIR motion
algorithm	PDR, fingerprint for initial position, particle filtering	self calibration, path loss, trilateration, temporal allignment	DOA of acoustic signals	self calibration, TDOA of acoustic signals	for each set of binary events iteratively looks for the minimum number of sensors that satisfy the condition	PDR, RFID RSSI fingerprinting fed to a genetic algorithm	finds the uncertainty zone of a person by a circle around them with radious of (their speed) × time	Use optimized sensor placement of PIR motion sensors to detect location from binary events