EECS545 Final Project Report
Indoor Location & Navigation

Jiaxing Yang
Computer Science Engineering Department
University of Michigan
Ann Arbor, MI 48105
yjxqwed@umich.edu

Wentao Zhang
Computer Science Engineering Department
University of Michigan
Ann Arbor, MI 48105
zwtao@umich.edu

Bohao Zhang
Robotics Institute
University of Michigan
Ann Arbor, MI 48105
jimzhang@umich.edu

Yuxin Chen
Robotics Institute
University of Michigan
Ann Arbor, MI 48105
chyuxin@umich.edu

Abstract

Modern buildings are becoming more and more complex. Accurate indoor location service is imperative to help people navigate inside a building. However, traditional outdoor navigation methods like GPS perform poorly inside a building due to the weak signal. We propose a method to address the indoor localization problem using machine learning on the data provided by a smartphone. Our approach uses kNN on the iBeacon data to estimate the floor and reach the accuracy of 64.7% if allowing ± 1 error. Besides, our method performs a kernel ridge regression on the IMU data to predict the position. It calibrates the estimation periodically with signal fingerprinting methods to eliminate the accumulated IMU data error. Our method achieves a position estimation error under 1m.

1 Introduction

The smartphone goes everywhere with people whether driving to the grocery store or shopping for holiday gifts. With users' permission, apps can use the location to provide contextual information. We can get driving directions, find a store, or receive alerts for nearby promotions. These handy features are enabled by GPS, which requires outdoor exposure for the best accuracy. Yet, there are many times when people are inside large structures, such as a shopping mall or event center. Accurate indoor positioning, based on public sensors and user permission, allows for a great location-based experience even when people are not outside.

Current positioning solutions have poor accuracy, particularly in multi-level buildings, or generalize poorly to small datasets. Additionally, GPS was built for a time before smartphones. Today's use cases often require more granularity than is typically available indoors.

Therefore, predicting the indoor position of smartphones based on real-time sensor data will not only improve the accuracy of GPS-based localization techniques, but also provide an alternative for indoor positioning.The sensor data consists of dense indoor signatures of WiFi, geomagnetic field, iBeacons etc., as well as ground truth positions collected from hundreds of buildings in cities. The IMU (accelerometer, rotation vector sensor) and geomagnetic field (magnetometer) readings collected by the smart-phone will also be recorded. With all these sensor data and a machine learning based positioning method, smartphone user can get a reliable indoor-localization service with com-

Preprint. Under review.

parable accuracy as outdoor, which will improve the indoor experience and reduce the large demand of mall maps for each specific building.

The paper is structured as follows. In Section 2, we estimate the floor number and the position at one certain floor separately. we proposed a kNN algorithm using iBeacon for indoor floor estimation. In Section 3, we applied three different regression methods and compare their performance. It turns out that the ANN using GPS and IMU achieve highest indoor position estimation accuracy. In Section 4, we gave a brief conclusion on the project as a whole.

2 Floor Prediction

2.1 METHODOLOGY

K-Nearest Neighbor (KNN) is a simple method that tries to perform classification by calculating the distance between features. The values of RSSI fingerprints depend on the physical distance between the iBeacon devices, which are labeled by UUID and MainID, and user mobile phone. The KNN algorithm considers K calibration points. The selection of these points based on selecting the closest K points in the feature space to approximate the position of the user.

The KNN algorithm starts by calculating the P-norm of three dimensions iBeacon information tuples (UUID, MainID, RSSI) (x_i) , where x_i belongs to the fingerprint radio map $x_i \in \mathbb{R}^N$.

The KNN algorithm calculate the distance between the measured iBeacon tuple and the tuples in matrix of iBeacon information as x_i, shown in the following equation

$$d(y - x_i) = (\sum_{j=1}^{|y|} |y_j - x_{ij}|^p)^{1/p} \tag{1}$$

where $d(\cdot)$ is the distance measured, x_{ij} is the j-th train tuple in class i. In case of $p = 1$ represents Manhattan norm-distance and in case of $p = 2$ represents using the Euclidean norm-distance. In this paper, the Euclidean distance is used. The algorithm selects the class which has the minimum Euclidean norm-distance with y as the predicted floor level.

2.2 PROPOSED METHOD

2.2.1 Pre-train

We notice that the amount of data is distributed unfairly among the floors. The floor 1-4 contains 77% of the total amount of data, while in the ideal case, they should represent 50% of the them. Thus, the data balance technique is applied to the data set. The data are replicated by $max(floor\_data\_num)/floor\_data\_num$ times. The figure 1 shows the data distribution before and after the data balancing.

2.2.2 Train

Our training set is composed of 12,000 trace files. IBeacon information tuples (UUID, MainID, RSSI) are extracted to form a matrix in a dimension of $N \times 3$. Then the UUID and MainID are transformed to One-hotpot representation. The onehot encoder is saved for future prediction. Then, the transformed array are used to train a KNN model with K = 8. The accuracy of the trained KNN model for the single iBeacon tuple is 44%

2.2.3 Prediction

The floor number will be predicted based on the trace file provided by the user. The iBeacon information tuple (UUID, MainID, RSSI) will be first extracted from the trace file. Then each tuple will be fed to the prediction model(K nearest neighbour), which will return a predicted floor for that tuple. And that floor will get one point. After all tuples have been processed, the floor with the highest score will be chosen as the predicted answer. The baseline prediction method gives 32.1% accuracy.

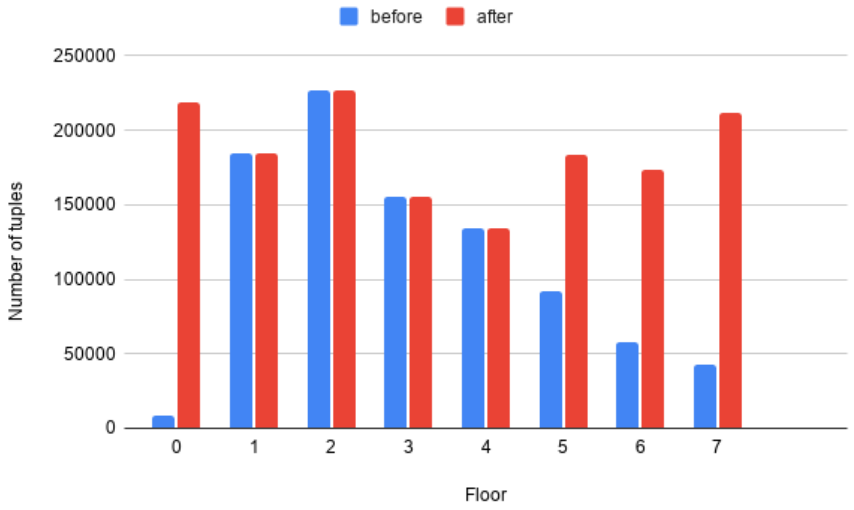


Figure 1: Demonstration on data sample distributions before and after data balance.

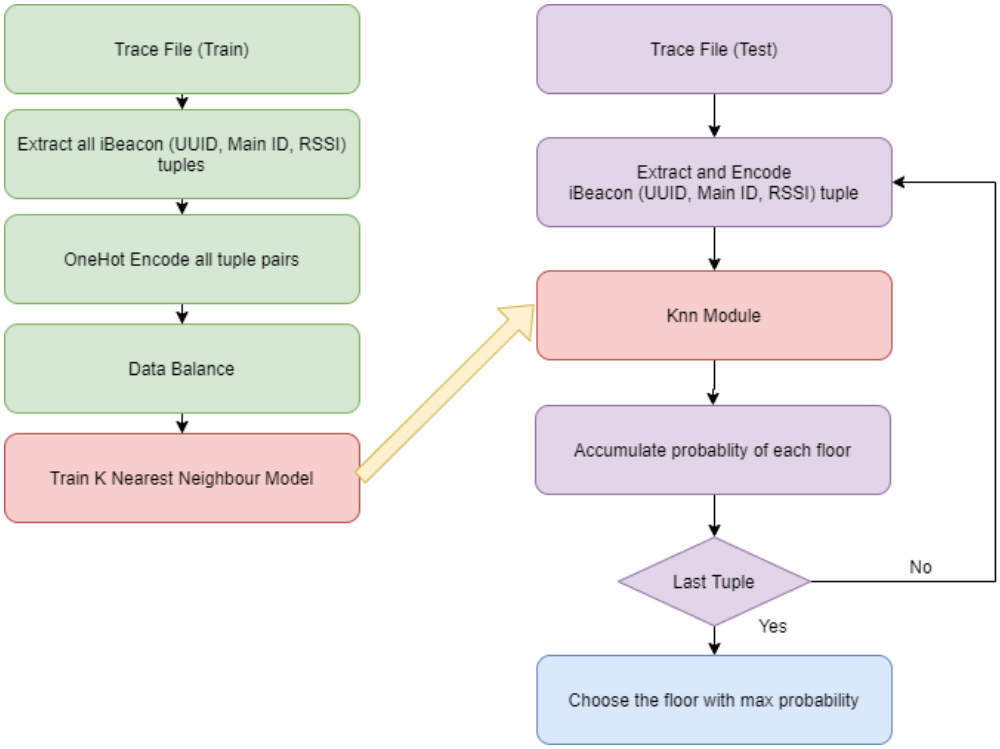


Figure 2: The flow graph of the Train and Prediction part of the proposed method. The trained KNN model is used to predict the probability of each floor based on the iBeacon information tuples. The floor with the highest probability is treated as the prediction result.

2.3 Probability Optimization

Instead of returning a predicted floor and vote that specific floor, the prediction model will return an array of probability of each floor and accumulate the probability across the information tuples. With the optimization, we found that the overall accuracy rises 2%. However, the accuracy for floor 3-7 rises by 26%. This shows that this optimization makes our method more tolerant with the noise.

2.4 Result Analysis

We test the model with 2,400 trace files and the test results for baseline and optimizations are shown in the table 1. The baseline implementation gives us 32% accuracy. The data balancing technique provides 12% accuracy. With the probability optimization, although the overall accuracy rises 2%, the accuracy for floor 3-7 rises by 26%.

We take a deep look at the miss prediction generated by the model. If we allow a floor tolerance, that is if the predicted floor is in the range of the correct floor ± 1, we will classify it as correct, then the overall accuracy rises to 64.7%, with the highest single floor (floor 6) accuracy to 90%. This shows that the noise in the data set, especially for floor 3-7, is large and our model can not reduce that noise further.

Floor	Baseline	Balanced	Balanced + proba	Balanced + proba + floor tolerance
0	0.03	0.73	0.60	0.63
1	0.78	0.72	0.70	0.72
2	0.90	0.82	0.60	0.64
3	0.06	0.04	0.12	0.42
4	0.48	0.20	0.24	0.56
5	0.06	0.44	0.50	0.72
6	0.08	0.32	0.40	0.90
7	0.06	0.36	0.46	0.58
Avg	0.321	0.439	0.447	0.647

Table 1: The test accuracy for floor prediction. With all optimization applied, 44.7% of accuracy is achieved. The floor tolerance boosts the accuracy to 64.7%

3 Position Estimation

3.1 Dataset Description & Process

The dataset given by (1) consists of many traces in different buildings recorded by the smartphone sensors. Specifically, the data that is useful for position prediction is twofold. The first part is the IMU sensor data which provides the information of motions of the smartphone, including acceleration (from three dimensions) and rotation vectors (from three dimensions) (2). The second part is the ground truth positions in the building frame provided by the host of the competition. The IMU sensor data is reported and collected in a high frequency (about 50 Hz). The ground truth positions are recorded in a low and not constant frequency. It is typically given when the smartphone makes a turn in a trace. Both parts of the data come with an unique timestamp which is measured by the smartphone timer.

A low pass filter implementation is given from the host to reduce the noise from raw IMU sensor data. A position interpolation implementation is also given to use both interpolation and filtered IMU data to compute a position estimation for any timestamp. It is required that the given timestamp should appear between two ground truth positions. Figure 3 shows an example of a trace in the dataset, including the ground truth positions and interpolated positions as an estimate of the actual trace.

We define the notation of the data for the discussion in the following sections. We use $t_i \in \mathbb{N}$, where $i \in \mathbb{N}$, to represent a possible time stamp coming together with a set of sensor data or ground truth position data. We use $a_i \in \mathbb{R}^3$ to represent the acceleration in 3-D world measured by smartphone IMU sensor at some time stamp t_i. We use $r_i \in \mathbb{R}^3$ to represent the rotation vector in 3-D world

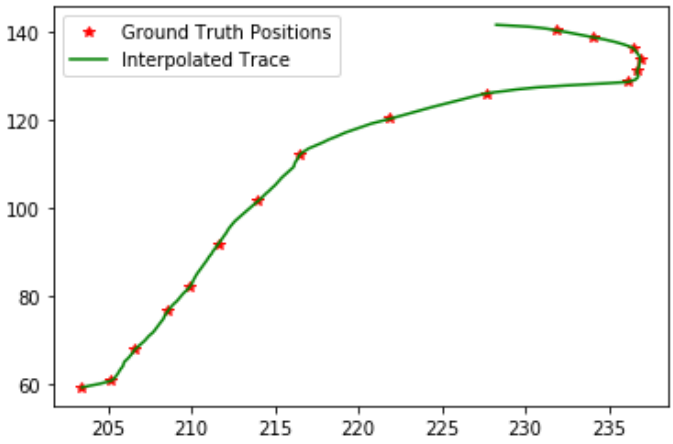


Figure 3: Demonstration on ground truth positions and their corresponding interpolated trace provided in the dataset. Red star points represent the ground truth positions directly provided in the dataset. The green trace consists of interpolated positions computed by the ground truth positions and filtered IMU sensor data.

measured by smartphone IMU sensor at some time stamp t_i. Both a_i and r_i refer to filtered sensor data. We use $d_i \in \mathbb{R}^2$ to represent the interpolated position in a 2-D map at some time stamp t_i.

3.2 Proposed Method

3.2.1 Framework Overview

Significant achievements on indoor localization in both academia and industry have been accomplished in the past two decades. Multiple means were surveyed to improve the accuracy of indoor localization, such as pure IMU sensor (3), ambience fingerprinting (4), FM fingerprinting (5) and GPS (6) as well. However, highly accurate smartphone-based indoor localization in a practical scenario is still an open problem (7). Typically, a pure IMU method will become less and less unreliable since the error in acceleration is accumulated all the time (3). While localization using signal fingerprinting may be time consuming and computationally expensive.

Here, we propose an machine learning based localization framework that fuses IMU sensor and the fingerprinting methods mentioned above to enable high frequency and computationally efficeint indoor position estimation without losing too much accuracy. It uses a simple predictor to estimate the change of position from IMU sensor data while compensating the accumulated error from IMU sensor with the aid from signal fingerprinting methods. The signal fingerprinting methods are called in a low frequency to save computation resource while the predictor based on IMU is called in a high frequency for every filtered IMU sensor data.

Our method is detailed in Algorithm 1. To initialize the algorithm, an initial position p_0 together with its corresponding IMU sensor data a_0, r_0 and time stamp t_0 is given. Since the outdoor GPS localization is accurate enough, we assume that p_0 is the GPS-estimated position, which is absolutely accurate, before a smartphone user enters a building and enables indoor localization service. We also defines a parameter T_{ref} as the period to call one signal fingerprinting method to calibrate current position estimated by IMU sensor. Every time when a period T_{ref} passes (Line 7), we call ComputeReferencePosition() function to calibrate current position p (Line 8) with a reference position. ComputeReferencePosition() refers to a selected signal fingerprinting method that generates reference positions. Otherwise, we use a Predictor() function to estimate the change of position (Line 14) based on filtered IMU sensor data from function ReadIMU() (Line 6) and update the current position by moving forward a step (Line 11). After that, the program reports a tuple of current timestamp t_i and current estimated position p. Note that T_{ref} is always larger than the IMU sensor frequency, which is equivalent to any $t_i - t_{i-1}$.

Algorithm 1 Proposed Method

Inputs: Initial position p_0, Initial time stamp t_0, Initial IMU data a_0, r_0
Parameters: Period to call signal fingerprinting method T_{ref}
Algorithm:

$p \leftarrow p_0 \triangleleft$ initialize current position
$\Delta p_0 \leftarrow$ Predictor(a_0, r_0)
$i = 1$
$t_{ref} = t_0 \triangleleft$ initialize timestamp when a reference position is provided
while isIndoor() **do**
  $\{t_i, a_i, r_i\} \leftarrow$ ReadIMU()
  if $t_i - t_{ref} > T_{ref}$ **then**
    $\{t_{ref}, p\} \leftarrow$ ComputeReferencePosition() $\triangleleft$ calibrate current position
    $t_i \leftarrow t_{ref}$
  else
    $p \leftarrow p + \Delta p_{i-1}(t_i - t_{i-1}) \triangleleft$ update current position
  end if
  Report(t_i, p)
  $\Delta p_i \leftarrow$ Predictor$(a_i, r_i) \triangleleft$ estimate change of position for next step
  $i \leftarrow i + 1$
end while

3.2.2 Change of Position Predictor Train & Evaluation

In the dataset, we assume that the interpolated traces are absolutely accurate and can represent the ground truth traces. Suppose there are N avaliable sensor data in the dataset. As a result, we consider the input as all acceleration and rotation vector data a_i and r_i at timestamp t_i for all $i = 1, \cdots, N$. For the output, we use the average velocity to represent change of position with respect to time. We first compute an interpolated position d_i for all the timestamps t_i. Then we use Equation 2 to compute the labels of the predictor.

$$v_i = \frac{d_{i+1} - d_i}{t_{i+1} - t_i},\ i = 1, \cdots, N-1 \tag{2}$$

Here $v_i \in \mathbb{R}^2$.

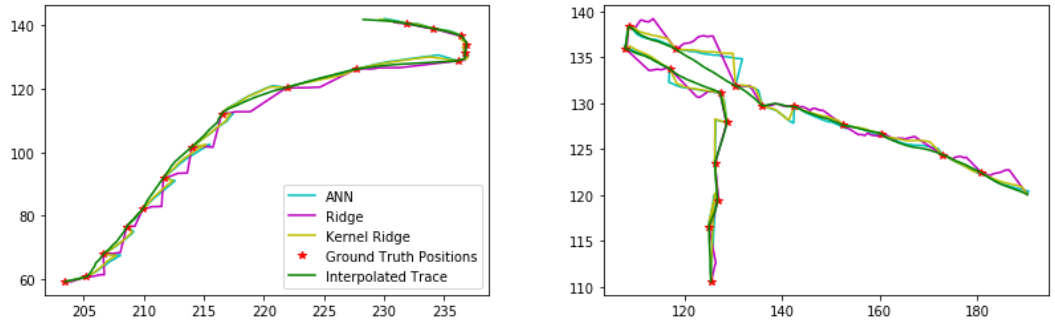
There are 73623 groups of data available which maps from a_i and r_i to v_i after processing. We randomly partitioned the data into train, validation and test set using the ratio of 60%, 20% and 20%.

We further apply regression analysis to the data to train a predictor model. Three regression methods are considered. The first one is a simple linear ridge regression. The second one is a kernel ridge regression with a polynomial kernel of degree 3. The third one is a three-layers neural network. All three layers are a combination of a linear layer with a ReLU layer. The dimension of all three linear layers is 500. Their performance are evaluated in Section 3.3.

3.3 Result Analysis & Comparison

We ran Algorithm 1 with three choices of predictor and plotted their estimated traces. Figure 4 provides a demonstration of the results of three predictors. Note that the estimated trace will be immediately reset to the reference position as soon as it is available after we call ComputeReferencePosition(). In the test cases, we treat the ground truth positions as reference positions and the interpolated traces as ground truth traces.

We can observe that the ridge method gives an inaccurate estimation of the change of positions, while the other two methods gives a better estimation. For most of the timestamps, the kernel ridge method gives a similar estimation compared with the neural network method. To further evaluate the performance of three given methods, we provide the norm difference between estimated traces and ground truth traces at every timestamp over the same four test traces in Figure 5.



(a) test trace 1

(b) test trace 2

(c) test trace 3

(d) test trace 4

Figure 4: Estimated traces using three different predictors over four test traces. The red points represent the ground truth positions in the test set. The green traces represent the interpolated trace given in the competition, which we treat as ground truth traces. The magenta traces represent the traces estimated using the ridge method. The cyan traces represent the traces estimated using an artificial neural network. The yellow traces represents the traces estimated using the kernel ridge method.

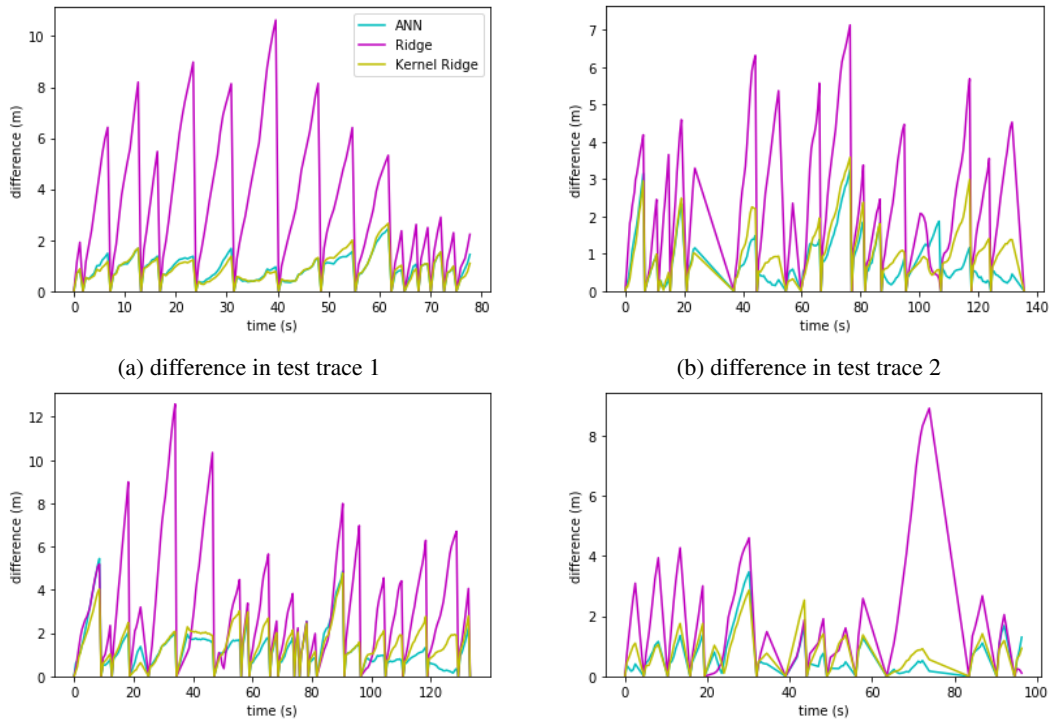
Table 2 shows the mean and standard deviation of differences of three methods over all timestamps in all test traces provided. The neural network we built is able to provide the most accurate estimations. The kernel ridge method, however, provides more stable estimations but with larger error. The mean error of both methods are within 1 meter.

Methods	Mean (m)	Standard deviation
Ridge Method	2.033262237516566	1.9404443522165973
Kernel Ridge Method	0.864887320767895	0.6506510708496157
Neural Network Method	0.7267895290771469	0.7263866152216657

Table 2: Mean and standard deviation of differences between estimated traces using three different predictors and ground truth traces at every timestamp over all test traces

4 Conclusion

Indoor location and navigation is a significant problem in modern life but is still challenging. We introduce a method that estimates both floor and position accurately boosted by machine learning. Our approach uses the smartphone-collected data in two aspects. It performs kNN on the iBeacon data for the floor estimation. Besides, it uses the IMU data and performs regression for the position estimation. With the two outputs, our method can give the final result for the indoor localization service.



(a) difference in test trace 1

(b) difference in test trace 2

(c) difference in test trace 3

(d) difference in test trace 4

Figure 5: Difference between estimated traces using three different predictors and ground truth traces at every timestamp over four test traces. The magenta data represent the difference using the ridge method. The cyan data represent the difference using an artificial neural network. The yellow data represents the difference using the kernel ridge method.

References

[1] MicrosoftResearch, "Indoor location & navigation." [Online]. Available: https://www.kaggle.com/c/indoor-location-navigation

[2] AndroidDevelopers, "Motion sensors." [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_motion

[3] F. Höflinger, R. Zhang, and L. M. Reindl, "Indoor-localization system using a micro-inertial measurement unit (imu)," in *2012 European Frequency and Time Forum*, 2012, pp. 443–447.

[4] M. Azizyan, I. Constandache, and R. Roy Choudhury, "Surroundsense: Mobile phone localization via ambience fingerprinting," in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '09. Association for Computing Machinery, 2009, p. 261–272.

[5] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha, "Fm-based indoor localization," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. Association for Computing Machinery, 2012, p. 169–182.

[6] M. Okamoto and C. Chen, "Improving gps-based indoor-outdoor detection with moving direction information from smartphone." Association for Computing Machinery, 2015.

[7] K. Liu, X. Liu, and X. Li, "Guoguo: Enabling fine-grained indoor localization via smartphone," in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '13. Association for Computing Machinery, 2013, p. 235–248.