Locating and Tracking BLE Beacons with Smartphones

Dongyao Chen[*], Kang G. Shin
University of Michigan, Ann Arbor
{chendy,kgshin}@umich.edu

Yurong Jiang[*], Kyu-Han Kim
Hewlett Packard Labs
{yurong.jiang,kyu-han.kim}@hpe.com

ABSTRACT

We present a smartphone-based application, called LocBLE, for enabling users to estimate the location of nearby Bluetooth low energy (BLE) beacons. In contrast to existing BLE beacon-based proximity applications that can only show coarse-grained (immediate, near, and far) distance estimation, LocBLE's fine-grained estimation can enhance human-environment interactions.

LocBLE has three salient features in estimating location from BLE beacon signals. First, it is adaptive to dynamic signal propagation environments by learning the environmental changes directly from the received signal strength (RSS). Second, it performs sensor-fusion for location estimation by utilizing motion sensor data and RSS readings from a smartphone. Finally, LocBLE improves location tracking accuracy with novel on-line calibration on a set of beacons nearby. We have built a prototype of LocBLE on smartphones and evaluated it on commodity proximity-enabled beacons. Our experimental results demonstrate that LocBLE achieves an average of 1.8m and 1.2m accuracies in locating indoor and outdoor BLE beacons, respectively.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing**;

KEYWORDS

Bluetooth low energy, Internet of Things (IoT)

1 INTRODUCTION

Bluetooth Low Energy (BLE) beacons have become very popular with numerous emerging applications of IoT (Internet-of-Thing), AR (Augmented Reality), and home automation. Proximity estimation, as one of the most representative features, is expected to expand human–environment interactions in various applications, including retail marketing [1], health-care [2], and transportation [3]. For example, BLE beacons have already been deployed in many retail giants, including Target and Macy's, showing the proximity of items on the customers' phones and thus creating a more engaging shopping experience [4].

CoNEXT '17, December 12–15, 2017, Incheon, Republic of Korea
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5422-6/17/12…$15.00
https://doi.org/10.1145/3143361.3143385

(a) Finding item (b) AR tagging

Figure 1: Featured use-cases of fine-grained location of BLE beacons.

Even though proximity beacons have been pervasively deployed, their functionalities are limited to a few "nearable" applications. Specifically, existing applications can only provide a rough proximity information.[1] This feature hampers the usability of BLE beacons in various scenarios, especially those requiring (2-dimensional) position information of beacons. Let's highlight two representative use-cases. Fig. 1(a) shows use of smartphones to find a lost item by locating an attached BLE beacon on the lost item, while Fig. 1(b) shows AR users' concentration on the items of their interest that are highlighted by an attached BLE beacon, thus helping them focus on items even when they can't see due to the blockage of line of sight.

The cause of limiting existing BLE beacon's usability is rooted at its low-power design. As we will elaborate in Sec. 2, to achieve the longest possible battery life, the BLE protocol design specifies a series of power-saving features, including low transmission power, narrow bandwidth, simplified connection design, and low duty cycle. Due to this ultra light-weighted protocol design, the received signal strength (RSS) of BLE beacon's advertisement becomes the only indicator for distance estimation. Unlike other popular location indicators, such as channel state information (CSI) [5, 6], time-of-flight, and angle-of-arrival, RSS readings are shown to exhibit large fluctuations due to dynamic propagation environments [6–8]. These features together make it challenging to locate and track BLE beacons.

To address this challenge, we present LocBLE, a low-cost smartphone-based application for estimating location of nearby commodity BLE beacons. In its core, the design of LocBLE is comprised of three novel components to mitigate RSS fluctuations and extract location information from limited resources.

First, since RSS reading is susceptible to environment changes, to make LocBLE adaptive to dynamic environments, we need to answer a challenging question "how to make LocBLE aware of

[*]The authors equally contributed to this work. This work was initiated during the first author's internship with Hewlett Packard Labs
[1]1-dimensional, four proximity zones (immediate, near, far, and unknown)

environmental changes?" To answer this question, we design a novel environmental estimation module, called EnvAware. This module infers environmental changes from RSS readings via a SVM-based algorithm. This estimation of environmental changes enables us to understand the varying channel condition, and allows LocBLE to adjust the following location estimation when the environment undergoes significant changes.

Second, with awareness of environmental changes, LocBLE needs to estimate the target beacon's location based on RSS readings. This task is challenging for two reasons: 1) the traditional log-based path-loss model of radio frequency (RF) signal is formulated for only calculating 1-dimensional distance [9]; 2) the parameters in the log-based model fluctuates due to different environments and hardware configurations. We address both challenges by introducing a novel data fusion scheme based on inertial sensor data and RSS readings on smartphones. To estimate location, our algorithm uses a reverse regression and only requires the user to make a short movement for measurements. To adapt to changing parameters, our algorithm estimates the parameter set, instead of using constant numbers. Unlike existing estimation schemes [10, 11], LocBLE employs a novel path loss exponent estimation based on data fusion of RSS readings and motion sensors for path loss exponent estimation on mobile devices.

Finally, to explore opportunities for refining LocBLE's accuracy, we propose a novel way to refine location estimation by locating multiple neighboring BLE beacons together. Our design is inspired by an observation that multiple BLE beacons are likely close to each other. For example, in a retail store, items of the same category are stocked together. We exploit this and propose a novel clustering algorithm for improving accuracy. Specifically, LocBLE uses an algorithm based on dynamic time warping (DTW), allowing further calibration of location estimation.

We have built a prototype of LocBLE on both iOS and Android platforms. We have evaluated LocBLE using commodity proximity BLE beacons in various (8 indoor and 1 outdoor) environments. LocBLE is shown to achieve an average of 1.8m indoor and 1.2m outdoor accuracies in estimating BLE locations.

This paper makes the following three contributions:

- Development of LocBLE, a system for estimating BLE beacon location. LocBLE runs on smartphones and is fully functional with commodity BLE beacons.
- Introduction of three key design elements for enabling LocBLE adaptable to various environments (Sec. 4, Sec. 5, Sec. 6);
- Implementation and evaluation of LocBLE on commodity smartphones and BLE beacons (Sec. 7).

2 BACKGROUND

2.1 Nearable Technologies

Recently, nearable technology has received significant attention for its potential for enriching human–environment interactions. Existing RFID, NFC tags, and BLE beacons have shown their capability of practical asset tracking. However, RFID tag tracking [12, 13] requires a dedicated RFID reader, making it infeasible on light-weighted devices (e.g. smartphones). NFC tags can be detected by smartphones, but their range is limited to tens of centimeters [14]. In contrast to these technologies, BLE beacons are compatible with commodity mobile devices, and have wider coverage (8–15m in indoor settings).

2.2 BLE Beacon Primer

The BLE beacon represents a class of BLE devices, and advertises its identifier to nearby devices by broadcasting a BLE signal. To enable both portability and long battery life, BLE beacons are usually powered by coin cell batteries and operate with a power-saving design to guarantee 1–3 years of lifetime. However, this ultra low-power design limits the resource that can be used for estimating the location of a BLE beacon.

Limited transmission power. BLE limits the transmission power to reduce energy consumption. BLE v4.0, v4.1, and v4.2 defined the maximum output power to be 10mW, which is 10x lower than WiFi transmission power specified by the FCC [15]. This feature constrains the BLE beacon's range (<15m indoor) and makes the BLE beacon signal more susceptible to path loss caused by the blockage of signal propagation [9]. Note that the upcoming BLE v5.0 sets the maximum output power to 100mW, but this high Tx power is designed exclusively for high power devices with Class 1 BLE chip [16].

Narrow bandwidth and frequency hopping. To coexist with WiFi and other RF signals that operate in the 2.4GHz frequency band, BLE specifications incorporate narrow bandwidth and frequency hopping. Specifically, in advertisement state, a BLE device hops in a fixed sequence of 3 dedicated channels (37, 38, and 39, each with 2MHz bandwidth). In connection state, the frequency changes pseudorandomly among 40 channels [17]. These features make the BLE signal more susceptible to frequency-selective fading [6].

Connectivity and advertisement of BLE beacons. According to BLE beacon's specification of connectivity [17], a connectable BLE beacon works as a BLE peripheral device (e.g., Bluetooth mouse) and can receive pairing requests, whereas a non-connectable beacon is essentially a BLE device that works only in broadcasting mode. To determine the connectivity of a targeted BLE beacon, the receiving device can inspect the connectivity type indicated by the first 4 bits in the header advertising channel protocol data units (PDUs). Interested readers are referred to the BLE specification (page 2567, [17]) for more details. The non-connectible mode of BLE beacons can extend battery life by limiting the interaction between the peripheral (e.g., BLE beacons) and central (e.g., smartphones that are scanning their surrounding beacons) devices. To reduce power consumption further, the duty cycles for broadcasting advertisement are limited to be under 100ms and 20ms on non-connectable and connectable beacons, respectively [18].

To be compatible with the low power design of existing commodity beacons, LocBLE focuses on locating non-connectable BLE beacons.

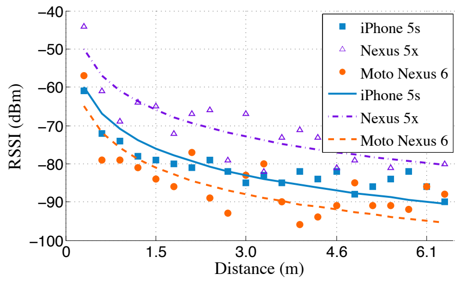


Figure 2: RSS reading on different smartphones.

2.3 RF Signal Fading

Like other RF signals, the BLE beacon signal suffers from signal degradation. Multipath fading occurs when RF signals reach the receiving antenna via multiple different paths. The different lengths of these paths make the received signals combined constructively or destructively. This effect further exacerbates the BLE signal's strength.

To cope with signal fading, existing WiFi and cellular networks use CSI feedback to adjust Tx power. Even though the manufacturer can replace the subset of an advertisement packet with a training symbol to enable CSI feedback, existing BLE beacons, such as iBeacon, EddyStone, and AltBeacon, are not compatible with this feature. For compatibility with existing BLE beacons without modification, we propose a new environment-aware method (Sec. 4) for adjusting the estimation of BLE beacon location.

2.4 RSS Measurement at Receiver

RSS measurements are also affected by the receiver's hardware configuration. Specifically, noises will be added to RSS readings due to the CMOS property of analog components, imperfections, and environment temperature. For example, the widely-used BroadCom BCM4334 WLAN/Bluetooth receiver chipset [19] has ±5 RSS accuracy at room temperature. This offset is another source of the noise in RSS readings. To mitigate this impact, we propose a novel energy-offset estimation scheme (Sec. 5).

2.5 RSS for Location Estimation

To evaluate the reliability of using RSS for BLE beacon location estimation, we first collect RSS readings on different smartphones in real-life indoor environments. In our experiment, we use 3 different smartphones, and walk away from the target BLE beacon on the same path. As shown in Fig. 2, despite the changes of data offsets on different smartphones, the RSS trend shows the same pattern. To mitigate the impact of RSS fluctuations further, we extract location information from the changing trend of RSS readings. Next, we will detail the design of LocBLE.

3 OVERVIEW

Fig. 3 shows the three-layer system architecture of LocBLE: *data collection*, *location estimation*, and *calibration*.

Data collection layer. This layer collects and processes various sensory data, including magnetometer, IMU, and RSS readings from BLE API (e.g., `CoreBluetooth` in iOS, `getBluetoothLeScanner` in Android).

Location estimation layer. This layer estimates the relative location of target BLE beacons. It has three main functionalities: 1) mitigating RSS fluctuations with both environment recognition (EnvAware) and adaptive noise filtering (ANF) (Sec. 4); 2) measuring the observer's movement by processing the motion sensor and magnetometer readings (Sec. 5); and 3) estimating the target's location by fusing measured RSS and motion data (Sec. 5).

Calibration layer. This layer explores opportunities for improving estimation accuracy. If there are multiple beacons with similar location estimation (or located nearby), their corresponding RSS readings often show a similar changing pattern. Based on this observation, the calibration layer first recognizes whether there are multiple beacons nearby, and then refine the location estimation with a probabilistic weight algorithm (Sec. 6).

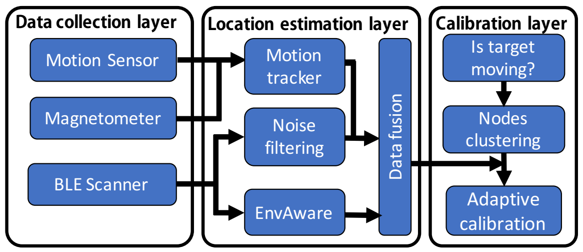


Figure 3: System architecture of LocBLE.

4 DATA PREPROCESSING BASED ON ENVIRONMENTAL CHANGES

The key reason for distorted RSS readings is the channel fluctuations caused by environmental changes. So, it is essential to preprocess RSS data for further analysis as we will show in Secs. 5 and 6. Specifically, LocBLE takes a two-step approach to preprocess RSS data: 1) recognizes environmental changes directly from RSS readings, and 2) adaptive noise filtering to smooth RSS data.

4.1 Environment recognition

Despite fluctuating RSS values, LocBLE uses the changing trend of RSS to estimate the target location. For a given model, our estimation will become the more accurate with more data. However, RSS readings may also be affected by environmental changes during the data collection process, thus yielding less accurate results. To address this problem, we propose an adaptive estimation method called EnvAware: it recognizes current environmental changes and use it to tune location estimation as discussed in Sec. 5.

Feature extraction and classifiers. Our RSS feature extraction segments the signal values into short (1–2s) windows and operates on them. Specifically, our feature vector comprised by the statistics of a new time window vector V: mean, variance, skewness. Beside these statistics, we also use 5 values directly from V: minimum, first quartile, median, third quartile, and max value. Finally, our feature vector is composed of the standardized 9 values described above. Such a feature definition turns out to be the most accurate for the various classifiers we tried: *SVM with various kernels*, *Decision Tree Classifier*, *RandomForest Classifier*, etc. In LocBLE, we chose *SVM with a linear kernel* as our classifier since it outperforms other algorithms in the ensemble.

To construct an RSS dataset by accounting for real-world signal propagation, we collected RSS data on smartphones in three representative environments: line-of-sight (LOS), partial-line-of-sight (p-LOS), non-line-of-sight (NLOS). p-LOS represents the propagation scenario that has blockage with a low blocking coefficient, such as glass, wooden door, and human body, etc., while NLOS represents the propagation scenario that has blockage with a high blocking coefficient, such as concrete wall, cinder wall, and metal board, etc. LocBLE aims to classify these three types of environment. Each data trace in our dataset was labeled with the corresponding environment. Our SVM-based algorithm is implemented by using sklearn module [20] in Python.

We collected training data in each designed type and labeled them according to the above 3 environment categories. For instance, for the blocked type, we placed one device behind a blocking object, the other device stores all the RSS data while moving around in front of the object. We also varied the blocking object, like wall, human body, etc. We used a time window of 2 seconds to generate the feature vector. Our classification achieved an excellent classification accuracy (94.7% precision and 94.5% recall for our three-type classification).

To incorporate EnvAware with our location estimation scheme (Sec. 5), LocBLE keeps monitoring environmental changes, and starts a new regression model only if new incoming data shows abrupt environmental changes.

4.2 Adaptive Noise Filtering

To smooth RSS data for next-step processing, LocBLE passes raw RSS data through an adaptive noise filter (ANF), and ANF is based on two noise filtering techniques: 1) a fine-tuned Butterworth filter, and 2) adaptive Kalman filter (AKF).

Butterworth filter design. To remove the effect of fast fading caused by environmental changes and device movements, we designed a low-pass filter based on a 6th-order Butterworth filter (BF).

Design of AKF. Our BF design can smooth fluctuating RSS data. However, the high order of BF also introduces delay and undermines the responsiveness of filtered data. To mitigate the impact of delay, we propose AKF, a modified Kalman filter. AKF enhances the responsiveness of filter by fusing raw RSS readings with BF output. More details can be found in [21].

Fig. 4 illustrates an example of BF + AKF processing results. BF achieves a much smoother result by filtering raw data, but it adds delay and is not fast in responding to RSS changes. We then apply AKF to achieve better performance than using BF alone.

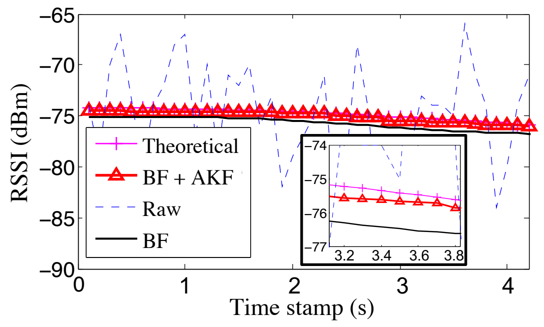


Figure 4: Performance of BF + AKF filtering design. The zoom-in figure shows a closer view

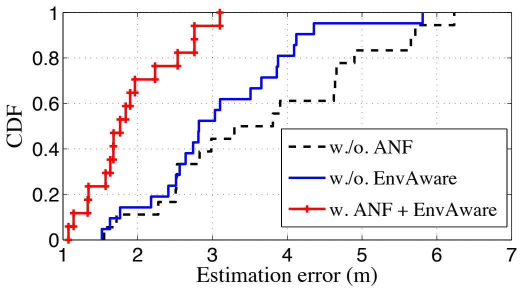


Figure 5: Performance of data preprocessing.

4.3 Performance of EnvAware and ANF

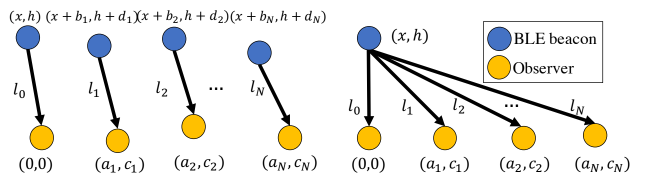
We evaluate the efficacy of EnvAware and ANF for LocBLE by testing their performance separately.

Performance of EnvAware. EnvAware ensures LocBLE to use the correct regression model for estimation. To quantify its effect on overall estimation accuracy, we compare LocBLE's performance with the case of removing EnvAware component. In particular, we tested performance in environments #2-#4 as shown in Table 1, such as the observer moves from behind the wall (NLOS) to line-of-sight (LOS) w.r.t. the target; people randomly come in between during the observer's movement to form p-LOS paths. We plotted the results in Fig. 5 (a). The removal of EnvAware is found to increase median error by more than 1m, because LocBLE adapts itself to environmental changes and updates the regression model accordingly, thus achieving better accuracy.

Performance of ANF. We used the same data from the EnvAware experiment, and plotted the performance of removing ANF in Fig. 5. We observed the accuracy degradation of more than 1.5m due to the absence of ANF, i.e., ANF plays a critical role in improving LocBLE's accuracy. This is because ANF's smoothing mitigates the impact of low channel coherence time due to user movements and environmental changes [9].

5 LOCATION ESTIMATION

LocBLE estimates the target beacon's location by using a regression-based data fusion of RSS data and motion sensor data. For location



(a) Locating a moving target (b) Locating a stationary target

Figure 6: Two different use-cases.

estimation of a stationary target, *e.g.,* finding a lost item with a BLE beacon, LocBLE performs estimation directly in the observer device.[2] For location estimation of a moving target, *e.g.,* locating a moving smartphone with the BLE beacon function turned on, LocBLE requires data transmission between the target and the observer. LocBLE gives users options to choose which mode they need.

Problem formulation. We first introduce our model from a general use-case in which both the observer and the target move randomly as shown in Fig. 6(a). We assume a coordinate plane with the origin of the observer's starting point, and x-axis as the observer's starting direction. Our goal is to estimate the target's relative location *(x,y)* in this coordinate. With accurate tracking of the movement for both the observer and the target, we have a_i, c_i as the observer's real-time x- and y-axis movements at the same time, the target's relative x- and y-axis movements as b_i, d_i, where $i \in [0, N]$, N is the total number of sample points. Note that $a_0 = 0, b_0 = 0, c_0 = 0, d_0 = 0$. Then, the corresponding distance $l_i = \sqrt{(x + b_i - a_i)^2 + (h + d_i - c_i)^2}$.

Using the path-loss model [9], we combine movements with RSS readings based on:

$$\begin{cases} RS = \Gamma(e) - 10n(e)\log(l_i) \\ l_i^2 = (x + b_i - a_i)^2 + (h + d_i - c_i)^2. \end{cases} \tag{1}$$

We modified the legacy log-based model to derive the first equation of Eq. (1). The key idea is that some parameters may vary with environment, and hence we use variable e to denote environmental changes. Here $\Gamma(e) = P + X(e)$, P denotes the power offset that depends on hardware configuration, $X(e)$ is the environment noise; $n(e)$ is the fading coefficient that varies with the environment.

To simplify the notation, we let $\epsilon = \exp\left(\frac{\Gamma(e)}{5n(e)}\right)$, $\eta = \exp\left(\frac{-1}{5n(e)}\right)$ and $p_i = b_i - a_i, q_i = d_i - c_i$, we can reformulate the second equation of Eq. (1) to:

$$p_i^2 + q_i^2 + 2xp_i + 2hq_i + x^2 + h^2 = \epsilon\eta^{RS_i}. \tag{2}$$

Note that Eq. (2) shows a similar form to an elliptical regression problem. So, we form a standard elliptical equation as:

$$Ap^2 + Bq^2 + Cp + Dq + G = \rho, \tag{3}$$

where $A = \frac{1}{\epsilon}, B = \frac{1}{\epsilon}, C = \frac{2x}{\epsilon}, D = \frac{2h}{\epsilon}, G = \frac{x^2+h^2}{\epsilon}$ and $\rho = \eta^{RS_i}$.

[2]In this paper, we define two types of devices: *observer* and *target*.

Our algorithm for deriving the above parameters is based on the least square regression; specifically, we have

$$P = (X^T X)^{-1} X^T Y, \tag{4}$$

where $P = [\mathbf{1}, A, B, C, D, G]'$ is the parameter vector, $X = [\mathbf{1}, p^2, pq, q^2, p, q]$ is the data matrix, and $Y = [\rho]$ is the output vector .

In case both the observer and the target are moving randomly, we assume the observer can communicate with the target. Specifically, after the measurement process, the target will send measurement data to the observer for processing. If the target remains stationary (Fig. 6(b)), the problem becomes much simpler and works in a standalone smart device. Specifically, q will become 0.

Solving for the fading coefficient. As discussed above, $n(e)$ cannot be derived explicitly, because the output variable η also contains $n(e)$. So, LocBLE determines $n(e)$ numerically by finding $\hat{n}^*(e)$:

$$\hat{n}^*(e) = arg \min_{\hat{n}(e)} (\mathbb{L}(\hat{x}, \hat{h}) - \mathbb{R}(\hat{n}(e), \Gamma(e)))^2, \tag{5}$$

where $\mathbb{L}(\cdot, \cdot, \cdot)$ and $\mathbb{R}(\cdot, \cdot)$ are the left- and the right-hand side formula of Eq. (2), respectively.

By solving the corresponding equation, we can estimate the location for both the target's stationary and moving cases. Thus, we can easily infer $\hat{x}$ and $\hat{h}$ from the parameters derived from the elliptical regression.

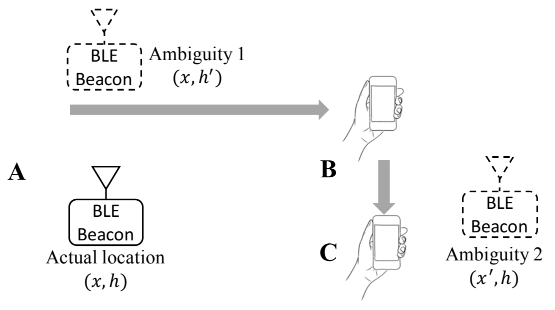


Figure 7: L-shaped movement for locating BLE beacons.

Estimation confidence. Let's revisit the signal propagation model $RS = \Gamma(e) - 10n(e)\log(d)$. With the actual/estimated coefficients $n(e)$ and $\Gamma(e)$, we can calculate the noise δ_{RS} for every RSS sensing point by subtracting the estimated RSS ($\hat{RS}$) from the original RSS (RS), i.e., $\delta_{RS} = RS - \hat{RS}$. Ideally, δ_{RS} follows a Gaussian distribution with 0 mean. However, in reality, δ_{RS} will not have 0 mean. Assume δ_{RS}'s mean and standard deviation (std) are μ and σ, respectively. Mathematically, Gaussian distribution's σ is robust to the change of its mean, so we assume σ remains the same and the original Gaussian noise follows $N(0, \sigma)$. Therefore, we treat $P(\mu)$ as a probability (estimation confidence), where $P(x)$ follows $N(0, \sigma)$.

5.1 Handling Symmetry Ambiguity

Due to the square root process for x and h, our scheme will generate two possible solutions which are symmetric to the moving path of the observer. To address this confusion, we design a simple moving pattern to follow for the observer to rule out ambiguity. LocBLE focuses on a 2-dimensional space.

L-Shaped Movement Design. Fig. 7 shows an example of 2-D movement we designed. Suppose the actual location of the target BLE beacon is at the right-hand side of movement direction $\overrightarrow{AB}$. The observer starts to move from point A to point B, using our location estimation algorithm, one can obtain a result set, $\{(x, h), (x, h')\}$, where (x, h) is the actual location, and (x, h') is the ambiguous location at the left-hand side of $\overrightarrow{AB}$ (*ambiguity 1* in Fig. 7). To overcome the ambiguity, the observer continues moving in a different direction, e.g., from B to C. With this movement, our algorithm can generate another result set, $\{(x, h), (x', h)\}$, where (x', h) is the other ambiguity at left-hand side of $\overrightarrow{BC}$. Thus, to get the location estimation of the actual beacon, we combine the two estimations together to disambiguate the false estimations. Specifically, we calculate the overlap of two result sets. In case both the observer and the target move, LocBLE only requires the observer to move with this pattern. We will later discuss how LocBLE estimates short-range movements.

5.2 Device Motion Estimation

LocBLE utilizes motion sensors on smartphones to measure the moving direction and distance in real-time. To make our motion tracker independent of phone postures, we use the well-known coordinate alignment [22] for transforming phone coordinate to earth coordinate.

To measures the dimension of L-shaped movement, LocBLE firstly measure the moving distance by counting steps recognized accelerometer readings for step count. For measuring turning, LocBLE uses gyroscope and magnetometer for tracking the degrees of turn.

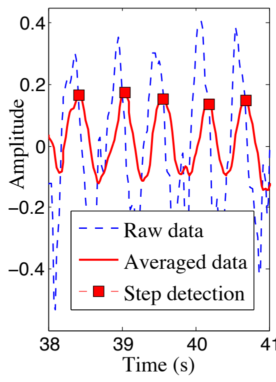
5.2.1 *Measuring moving distance.*

Measuring the user's movement with a smartphone has been studied extensively [23–25]. To determine the moving distance, we first use the accelerometer to detect step, and then combine it with step length to get the walking distance. Our step counter first smoothes the accelerometer data by using the moving average filter, then uses a voting algorithm to detect the peak, which represents the middle status of one gait cycle. The performance of our step counter is plotted in Fig. 8(a).

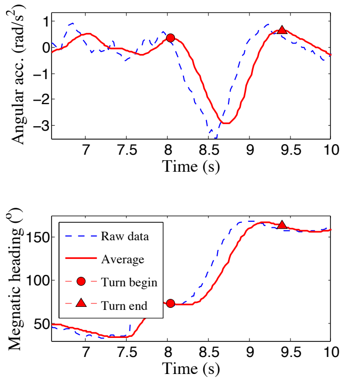
To infer the moving steps, we use a similar rationale as in [26]. Specifically, we can infer step length by inspecting the step frequency.

5.2.2 *Measuring turning angle.*

To measure turns, we first analyze gyroscope to identify turning behavior, then use magnetic heading to infer a specific turning angle. The magnetic field reading is known to fluctuate in indoor environments, but it is accurate over a short period time [23]. Specifically, to identify turning behavior, our turn detector inspects gyroscope readings to identify the bump caused by the turning behavior. Our algorithm can accurately track the beginning and ending points of a bump. Then, we find the corresponding points in the magnetic heading to get the turning angle, as shown in Fig. 8(b).



(a) Step detection



(b) Turn detection

Figure 8: Step and turn detection in LocBLE.

Our experimental results show that the accuracy of step-based moving distance estimation is around 94.77%, and the average angle estimation error is 3.45°. To further increase the measurement accuracy of the L-shaped movement in real-world applications, LocBLE can avoid the turning angle measurement step by explicitly asking the user to make a right angle (angle of 90°) turn when s/he makes an L-shaped movement.

Algorithm 1 Relative Location Estimation Algorithm

```
1: INPUT: $rs_o$ is the observer received RSS, $motion_o$ is the observer
   motion, similarly ($rs_t$, $motion_t$) for the target
2: Function EstimateLocation ($rs_o$, $motion_o$, $rs_t$, $motion_t$)
3: for every time window in $rs_o$ do
4:     if The Target is Stationary then
5:         Detect the observer's movement (step and direction)
6:     else
7:         Detect both the observer's and the target's movements
8:     Match the movement to RSS data with timestamp
9:     Classify its environment with EnvAware and filter the noise with
       ANF
10:    if Environment hasn't change then
11:        Continue the regression by appending the data
12:    else
13:        Start a new regression with the data
14:    Update target location estimation and probability
15: return Target location estimation and its probability
```

5.3 Summary of Location Estimation

Algo. 1 summarizes our data preprocessing and location estimation. At input, we collect a new data batch every 2–3 seconds with approximately 20 RSS samples per data batch, and motion data is also recorded in another data batch. With input data, LocBLE starts a 3-step estimation process. First, it determines the target's status and estimates the observer's (and the target's if moving) step and direction. Second, LocBLE matches the movement with the RSS data based on the timestamp. Based on the environment classification, LocBLE turns to the regression by appending the new data batch if the environment remains unchanged, or to start a new regression with the data if the environment changes. LocBLE then

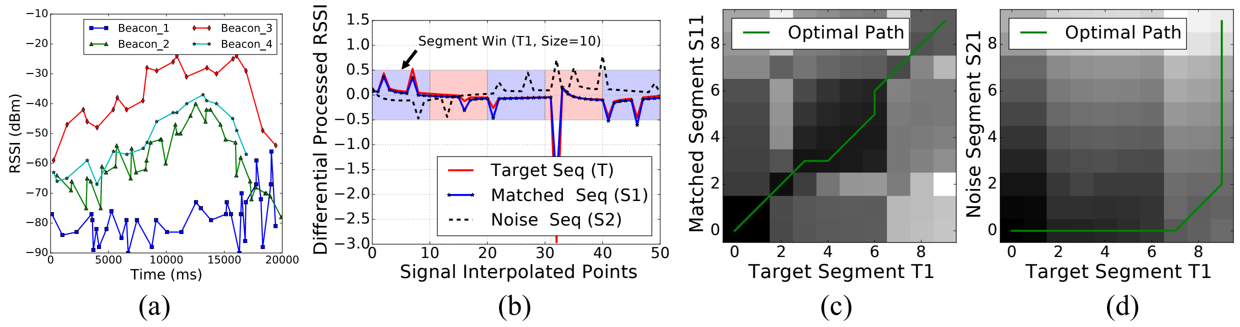


Figure 9: (a) Sample RSS sequences of 4 beacons; (b) Illustration for segmentation of 3 BLE sequences; (c) DTW cost matrix for successful matching; (d) DTW cost matrix for unsuccessful matching.

applies the adaptive noise filtering. Finally, we return the location estimation with the corresponding probability by combining these probabilistic estimations.

6 MULTI-BEACON CALIBRATION

Due to the low cost (usually \$6–15 each), the deployment of BLE beacons can be very flexible. Here we will show LocBLE is capable of refining the localization result with awareness of neighboring beacons. Specifically, LocBLE will first determine whether or not other neighboring beacons can form a cluster. If such an opportunity exist, LocBLE will calibrate the estimation result based on RSS readings from multiple clustered beacons. This method enables the user to get more accurate result within a *single* measurement.

6.1 Clustering Correction at Target

Neighboring devices are usually unknown to each other beforehand. LocBLE aims to group them based only on BLE beacon signals. To test how RSS signal pattern can be altered by physical layout, we firstly inspect real-world RSS measurement results of 4 beacons. Fig. 9(a) shows RSS sequences of one measurement, in which beacon 4 is the target device (5m away from observer). Beacons 2 and 3 stay close (0.3m between them) to the target and beacon 1 is not nearby (4m away). Beacons 2 and 3 are shown to exhibit a similar pattern of RSS changes. Since our measurement usually requires the observer to make an "L-shaped" movement, such a pattern becomes unique in geometry. According to our experimental results, beacons exhibit disparate RSS trends when they are placed far apart from each other. Moreover, the target RSS frequency is found to drop from 8Hz to around 3Hz due to interference. These observations led us to propose an algorithm based on dynamic time warping (DTW) [27] to capture the unique trend of RSS readings and cluster beacons 2, 3, and 4 together. DTW is, in general, used to align and measure the similarity between two temporal sequences of data. It formulates the cost matrix based on Euclidean distance between two datasets and then picks the path with the lowest cost as the alignment [27].

There are three challenges in using DTW to accurately to match sequences with the target sequence: 1) different devices may have different sensor sampling frequencies and may sense different signal amplitudes due to chipset and environment variations; 2) DTW has high time complexity ($O(n^2)$), which is not suitable for large data sequence matching; 3) possible significant noise in a data sequence may lead to incorrect matching for the data sequence. To deal with these challenges, LocBLE uses a fixed window DTW voting algorithm.

First, to deal with device heterogeneity and environment diversity, our BLE signal processing algorithm filters out high-frequency noises, and then differentiates the RSS sequences to avoid using absolute values.

Second, we reduce the computation overhead by dividing a large data sequence into small data segments and utilize the lower bounding technique [28]. Fig. 9(b) describes our algorithm for beacons 4, 3, 1. We assume target beacon 4 sequence T and two other sequences beacon 3 as $S1$ and beacon 1 as $S2$. We first divide T into equal-length segments. We found the segment length of 10 to yield the best tradeoff between accuracy and computation complexity. We then split the other candidate sequences according to T_i's timestamp, and interpolate them to match T's segments.

In Fig. 9(b), the data segments are illustrated with red and blue shaded boxes. Each segment is validated with the lower bounding technique. Basically, we create a bounding envelope above and below each target segment using the warping window. Then, we obtain the squared sum of the distances from every part of the candidate segment not falling within the bounding envelope, to the nearest orthogonal edge of the bounding envelope, and check if it is less than a threshold. The empirical threshold we set for the segment of 10 data batch is 6.1, which is the same as the DTW similarity threshold. Only valid data segments can be passed to the DTW similarity matching. This lower bound testing is 100x faster than DTW computing for the same size data according to our experiments. We show the matching for first segments in Fig. 9(b), Fig. 9(c) (successful matching) for beacon 3, and Fig. 9(d) (unsuccessful matching) for beacon 1. Our scheme is found to be at least 2x faster than applying DTW directly to the original sequence.

Finally, to obtain the final matching result, we use straightforward voting. For every BLE RSS sequence, each segment will be determined to be successful or unsuccessful based on the above DTW matching. We then determine a BLE RSS sequence to match successfully by checking more than a half of the sequence's segments to match the target segments.

6.2 Calibration

We calibrate the target position using the candidate estimations with probabilistic weights. Note that every target position estimation comes with a probability weight (Sec. 4), and our calibration works with this estimation. Suppose there are N candidate target positions with probability p_i for each candidate i, then we use the normalized weight $\frac{p_i}{(\sum_{j=1}^{N} p_j)}$ and compute the weighted sum of the candidate positions as the final estimation. The detail of our algorithm can be found in Algo. 2.

Algorithm 2 ClusteringCalibration Algorithm

```
1: INPUT: Pre-processed LocBLE's RSS sensing sequences for different
   devices $S_1, \ldots, S_N$ and observer–target sequence $T$ , $m_o$ is the ob-
   server's motion and $m_t$ is the target's motion
2: Function ClusteringCalbration ($S_1, \ldots, S_N, T, m_o, m_t$)
3: Init $Target\_Positions$ as an empty list
4: Split $T$ into segments $T_1, \ldots, T_m$ with 10 points each
5: for every sequence $S_i$ do
6:     Split $S_i$ into segments $S_{jk}$ ($k$ in (1, $m$)) according to $T_i$
7:     Init $matched\_count = 0$
8:     for every pair $S_{ik}$ and $T_k$ ($k$ in (1, $m$)) do
9:         if $S_{ik}$ passes lower bound threshold and $S_{ik}$ satisfies DTW simi-
           larity threshold then
10:            $matched\_count$ ++
11:    if $matched\_count > \frac{m}{2}$ then
12:        Set $target\_pos = EstimatePosition(rs_i, rs, m_o, m_t)$
13:        Add $target\_pos$ to $Target\_Positions$
14: Compute the weight $w_i$ for each $target\_pos_i$ in $Target\_Positions$
15: return Weighted Sum of $Target\_Positions$
```

7 EVALUATION

This section provides the details of our implementation and experiment setting, a demonstration of using LocBLE in action, evaluation of LocBLE's performance and overhead in various environments.

7.1 Implementation

We have implemented LocBLE on both Android and iOS devices. LocBLE runs on recent generations of iOS (version iOS 10.2.1) and Android (Android 6.0) devices, including iPhone 5s/6/6s/7, iPad Air, Nexus 6P. For iOS, LocBLE relies on *CoreBluetooth* framework to obtain the BLE RSS data and *CoreMotion* framework to get device sensing information. Since iOS 5.0, Apple no longer allows an iOS device to read unknown Bluetooth device's UUID, LocBLE requires the user to add the target device UUID to the app. It uses library SWIX [29] for the regression and machine learning classifier. For Android, we use the Bluetooth package for BLE scan and sensor package for motion sensing data. For the moving target scenario, we use UPnP [30] protocol to enable direct communication between two devices.

As shown in Fig. 10(a), LocBLE's GUI contains both *measure* and *navigation* modes. In *measure* mode, the user selects the target device and then follows the LocBLE's instruction arrow for an L-shape move. The instruction arrow grows based on the step detector, while LocBLE estimates the user movement with sensor data. Upon completion, LocBLE computes and shows the relative position of the target device. In *navigation* mode, LocBLE provides instructions based on the measured target position so that the user can find the target device. In LocBLE's current implementation, navigation is based on standard dead-reckoning with a step counter [31].

7.2 Methodology and Metrics

Experimental Setup. Our evaluation focuses on iBeacon devices, including various models of off-the-shelf smart devices and dedicated BLE beacons, including iPhone 5/5s/6/6s, Estimote beacons [32], and RadBeacon USB dongle [33]. We configured the BLE beacons to broadcast at 10Hz in order to be consistent with sampling rate. In our experiments, we assumed the static targets except for the moving target experiment. Our experiment setting includes various real-life environments. As shown in Table 1, our experimental setting includes 9 different environments (indexed from 1 to 9) — 8 indoor and 1 outdoor environments. The settings also include line-of-sight (LOS) and non-line-of-sight (NLOS) scenarios in which direct paths are blocked by furniture, store/shop racks, and human bodies. To test the practicability of LocBLE, our indoor test environment did not rule out WiFi access points.

Metrics and Ground Truth. The metric of our interest is the target location estimation error, or the difference in distance between the target's estimated location and the ground truth. In the case of a moving target, we measured the target location estimation error at its initial location. In order to accurately compare LocBLE's optimization algorithms against other alternatives, we performed a trace analysis. For this analysis, we collected the traces in 9 different environments (Table 1) covering most of our daily use-cases. Each trace had distances ranging from 3 to 16m and around 40 BLE data samples. Our dataset has nearly 300MB over the distance measurements of 1,000m. This dataset is used to evaluate LocBLE as described below.

7.3 Demonstrate LocBLE in Action

Demo [34] shows an example process of LocBLE navigation functionality. We randomly place the target Estimote beacon in an office and then use LocBLE to measure the location and navigate to find the target device. We measured the distance from LocBLE's navigation destination to the actual beacon location as the overall error. We conducted 20 runs with the distance to the target device ranges from 4 to 12m and show the overall error CDF in Fig. 10(b). The median overall error is 1.5m, 75 percentile error is 2m and the maximum error is less than 3m. This experiment demonstrates the feasibility of measuring the target BLE device's location and

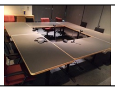
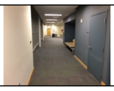
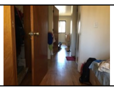
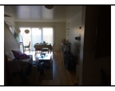
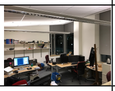
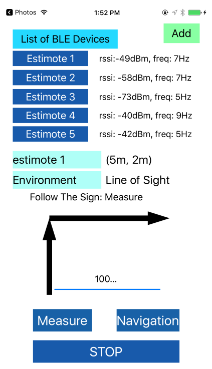
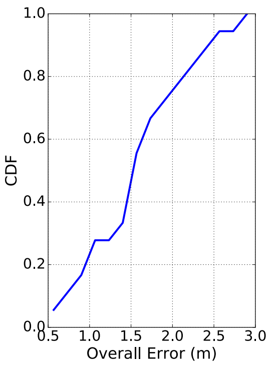
Index	1	2	3	4	5	6	7	8	9
Image									
Name	Meeting room	Hallway	Bedroom	Living room	Restaurant	Store	Labs	Hall	Parking lot
Scale	5×5 (m^2)	8×3 (m^2)	7×7 (m^2)	7×7 (m^2)	9×10 (m^2)	9×10 (m^2)	8×10 (m^2)	9×11 (m^2)	16×15 (m^2)
Acc. (m)	0.8 ± 0.2	1.4 ± 0.3	1.4 ± 0.4	1.6 ± 0.3	1.6 ± 0.4	1.8 ± 0.6	2.3 ± 0.5	2.1 ± 0.5	1.2 ± 0.5

Table 1: Details of experimental environments.



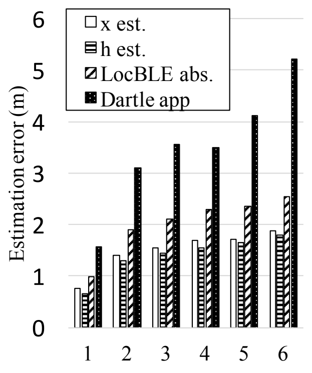
List of BLE Devices
Add
Estimote 1 rssi:-49dBm, freq: 7Hz
Estimote 2 rssi: -58dBm, freq: 7Hz
Estimote 3 rssi: -73dBm, freq: 5Hz
Estimote 4 rssi: -40dBm, freq: 9Hz
Estimote 5 rssi: -42dBm, freq: 5Hz
estimote 1 (5m, 2m)
Environment Line of Sight
Follow The Sign: Measure
100...
Measure Navigation
STOP

(a)

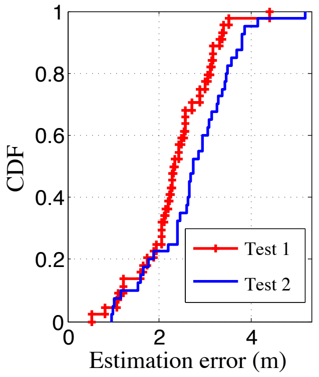


(b)

Figure 10: (a) LocBLE App UI; (b) overall error in action



(a) Stationary



(b) Moving

Figure 11: Performance of stationary and moving device scenarios. In (a), the x-axis shows the environment index

navigating to the target with the measured location, and handles various practical scenarios with reasonably good accuracy.

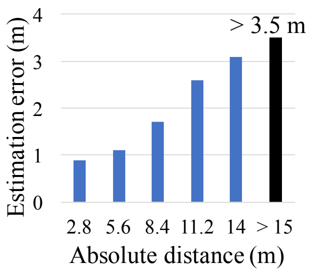
7.4 Performance Analysis of LocBLE in Different Environments

We now evaluate LocBLE's performance for both stationary and moving target devices in various environments.

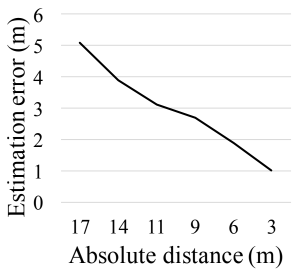
7.4.1 Stationary Target. A main application of LocBLE is to locate stationary BLE beacons. To guarantee the generality of our experiment, we tested LocBLE's performance in environments #1 – #6 as shown in Table 1. The original distance between the target and the observer in different environments are 4.5m, 6.4m, 6.7m, 6.8m, 9.1m and 7.9m, respectively. We plotted the mean error for x, h and absolute distance $\sqrt{(x^2 + h^2)}$ in Fig. 11(a). LocBLE is found to have less than an absolute distance error of 1m for the meeting room. In more challenging environments, LocBLE is found to have less than 2.4m absolute distance error due to the multi-path effects. LocBLE is found capable of providing the target's actual location estimation, i.e., x and h, while no existing solution has this capability. Instead, the existing solutions focus on range estimation with BLE proximity capability. So, we choose the best ranging app called *Dartle* [35] for comparison. We measured it under the same settings and plot its performance in Fig. 11(a) and find LocBLE achieving 30% less error than Dartle app's ranging estimation.

Performance analysis in different environments. Table 1 summarizes the LocBLE's performance from all ranges in locating stationary beacons. The results in the 5th row show LocBLE's mean accuracy with a 75%-confidence interval. There are two main takeaways from this study of LocBLE's performance. First, LocBLE shows better performance when a LOS path exists. The meeting room environment shows the best performance with <1m accuracy, while the store and lab environment show the worst performance due to the existence of highly reflective blocking objects (e.g., high market and server racks) in the signal propagation path. Second LocBLE shows stable performance with different environment settings where NLOS paths exist. Specifically, environments #2 – #6 show similar accuracies. As we will show in Sec. 4.3, our EnvAware helps LocBLE refine the estimation results when the environment changes.

7.4.2 Moving Target. Another feature of LocBLE is the ability to estimate the relative location of a moving target. To the best of our knowledge, there doesn't exist any solution for this purpose. In order to test LocBLE's performance for a moving target, we set up experiments with two users, one as the observer and the other as the target. Each user holds an iOS device with LocBLE activated. We fixed the start point for both users and pre-define several moving directions for each user. The two users moved in different directions

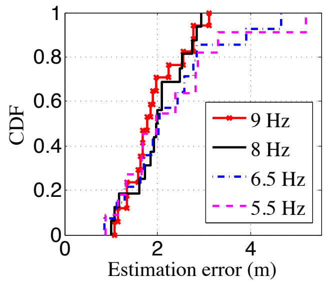


(a) Different target distance

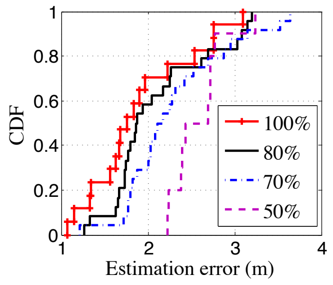


(b) Navigation performance

Figure 12: LocBLE's performance for different target distances and navigation



(a) Changing sampling frequency



(b) Varying data length

Figure 13: Comparison of estimation results for different sampling frequencies and data quantities.

and at different speeds. Moreover, both users move during the measurement. After measurement, the target transfers RSS and motion data traces to the observer for distance estimation. We tested the performance in environments #9 (test 1) and #8 (test 2), more than 40 experiments have been conducted in each environment. Test 1 changed distance from 3m to 9m, whereas in test 2 did from 3m to14m. We plot the CDF of error in Fig. 11(b). The results indicate the accuracy of less than 2.5m for more than 50% of data we collected. There are two sources of error affecting the performance. First, the BLE signal blockage changes too fast for the environment classifier to capture during the human's walking. Second, the error in movement estimation accumulates, especially in estimating the moving direction of two users. Thus, LocBLE can expect more accurate estimation with a better environment adapter and fine movement tracking.

7.5 Performance Analysis with Changing Target Distance

Target Distance on Measurement Accuracy. Due to the log-based degradation of RF signal, a natural question is: how does LocBLE's performance change with difference target distances? As discussed in literature [17], BLE-based sensing shows valid proximity estimation within an about 15m range. To ensure the same environment for a fair comparison, we conducted experiments in a sufficiently large outdoor parking lot. We collected the estimation results with LocBLE placed at 11 different testing points, two adjacent points of which were separated by 2.8m. For every point, we repeated the experiment 5 times and plotted the mean error in Fig. 12(a). LocBLE is found able to achieve around 1m accuracy within 5.6m and <3m accuracy within an 11.2m range. However, if the distance is over 14m, the performance degrades significantly to more than 3m which is consistent with our expectation. This is because a log-based propagation model has a minor decrease at long distances.

Target Distance on Navigation. LocBLE provides a navigation service for the observer to find the target. So, we assess LocBLE's navigation performance with a use-case. Basically, an observer who is 16.5m away from the target first estimates the target's location with LocBLE. Then, the observer follows the guidance of LocBLE to approach the target. We recorded the estimation accuracy at different locations. We repeated the experiments 3 times and show the mean error at different points in Fig. 12 (b). In the beginning, due to the long distance and insufficient BLE sensing data, the error reaches nearly 5m. LocBLE's performance improves as the observer approaches the target, especially when the distance reduces to 3m, the error drops to 1m.

7.6 LocBLE's Performance with Different Settings

In real-life applications, it is unlikely to keep settings ideal and homogeneous due to different hardware configurations and user behavior. Thus, it is necessary to study whether LocBLE will vary drastically with settings. Specifically, we focus on evaluating how sampling frequency, user's walking distance, and types of BLE beacon will impact on LocBLE's performance.

7.6.1 What sampling frequency is sufficient? Devices with different BLE chipsets and OSes may sense or broadcast BLE at different frequencies. For example, the sampling rate is 9Hz for iPhone 6s and 8Hz for Nexus 6P. We quantify the impact of sampling frequency on the estimation accuracy by re-sampling our data at a lower frequency. Our original data sampling rate is about 9Hz with iOS devices and the average length for every measurement are about 40 points. We conducted experiments in environment #2–#4, and compared frequencies 8Hz, 6.5Hz, and 5.5Hz (by inserting an idle delay between two consecutive scans). The result shown in Fig. 13(a) indicates that with a lower sampling frequency of data points being used for LocBLE, the medians of estimation results remain stable, but in the worst case, the lower sampling rate may degrade the performance, because a lower sampling frequency still ensures data completeness, but more susceptible to noises.

7.6.2 How far does the observer need to walk? Ideally, given an accuracy requirement, a shorter walk will yield better user experience. In LocBLE, we require the observer to walk 3.5–5m in total for an L-Shape path. So, how does the measurement of walking distance affect the overall estimation accuracy? To answer this

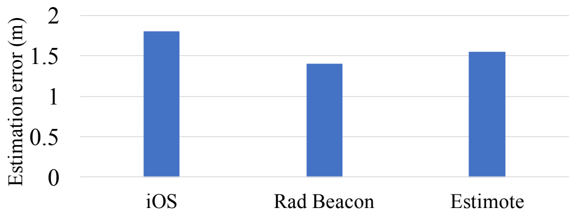


Figure 14: Performance on different beacons

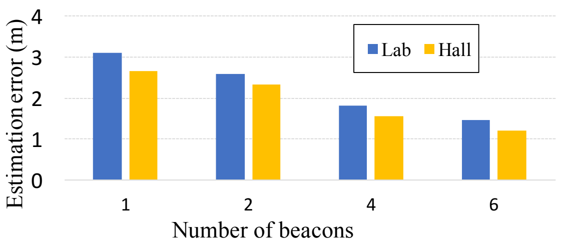


Figure 15: Calibration performance in two environments

question, we evaluated its impact by varying the length of the measurement data. The CDF of error is plotted in Fig. 13 (b). We find the performance remaining stable after reducing the data to 80% of the original data, but starts to degrade at 70%, and becomes much worse at 50%. This indicates that LocBLE needs at least 80% of data, that is ~3m walking distance, to capture the signal characteristics and provide accurate estimation. In practice, LocBLE asks the observer to take around 4–6 steps in the measurement, usually taking about 3–5s.

7.6.3 Does BLE beacon type matter? To verify this, we used three common types of BLE beacons in the market as the targets and measured the estimation errors in environment #2. Fig. 14 shows the average errors for using three different types of BLE beacons as the target. Dedicated BLE beacons (RadBeacon [33] and Estimote [32]) have slight advantages over smart devices integrated beacons, as the chips in smart devices are built more compactly. Regardless of this minor difference, the experimental results show that LocBLE doesn't depend on specific BLE devices.

7.7 Clustering Calibration

We evaluate the efficacy of our clustering scheme on enhancing accuracy. We conducted experiments in scenarios 7 and 8 as shown in Table 1 including a lab environment with a concrete wall block in the transmission path, and a hall environment with a construction in between. Due to heavy blockage and long distance, the accuracy for one target with a single beacon averages only 3m. As we increase the number of beacons, the accuracy improves as shown in Fig. 15. With 6 beacons, LocBLE reduces the error by half. A single device's estimation has a low estimation confidence and indicates low accuracy in these environments. That is, with LocBLE's clustering calibration, the estimations from those neighboring devices compensate the noise in the challenging environments, and improve the overall accuracy.

7.8 System Overhead

LocBLE doesn't require any hardware or OS modification and has low energy consumption at both the target and observer sides. To quantify this, we logged the energy consumption as a time series by instrumenting LocBLE on XCode [36]. The result shows that LocBLE increases CPU usage by 14% and energy consumption by 12%, which is slightly higher than the existing ranging app (Dartle [35]) which has 11.3% and 11% increases, respectively. We also expect similar energy consumption on Android.

8 RELATED WORK

8.1 Nearable Technologies

The BLE proximity beacon technology has received significant attention for its potential for enriching human–environment interactions. For example, manufacturers, such as Apple [37], Google [38] and open-sourced community [39], proposed their own BLE proximity beacon specifications. However, existing pervasively-deployed BLE beacons are only able to provide coarse-grained proximity.

LocBLE is a fundamental enhancement of existing nearable technologies by enabling estimation of the target beacon's location with a single smartphone, without modification to the smartphone and the proximity beacon. These features extend the usability of existing nearable beacons.

8.2 Transmitter Tracking with RF Signals

Transmitter localization based on RF signals has been an active area of research, achieving high indoor localization accuracy. In general, to locate the RF signal transmitter's location, existing research efforts focus on angle-of-arrival (AoA) [40, 41], time-of-flight (ToF) [5, 42], or RSS [43, 44] analysis. To measure AoA, the RF receiver needs to have an antenna array for analyzing phase differences. To measure ToF, the RF receiver should have a nanosecond-level sampling rate to achieve fine-granularity, which is infeasible on mobile devices. Thus, these techniques are infeasible for low-power BLE beacons.

RSS-based approaches, *e.g.,* Borealis [45], is designed for locating a WiFi AP in outdoor environments without dedicated hardware. However, [45] requires OS modification on smartphones to support WiFi RSS scan on a single channel. Unlike AoA, ToF and RSS based device tracking, FindMiMo [46] is a tracking system for finding a mobile phone by utilizing various logging information, such as AP identifier, GPS, and motion sensors. This system is infeasible to locate low-power BLE beacons due to the limited information the user can get from BLE beacon advertisements. In contrast with existing approaches, LocBLE does not require OS modification on smartphones or BLE beacons.

802.11 mc [47] enabled fine-grained ToF measurement at receiver side. However, this protocol is designed for WiFi devices, thus not feasible in IoT use-cases.

8.3 Environment Estimation

Path-loss exponent (PLE) estimation has been studied extensively in the area of wireless sensor networks. Unlike the various estimation schemes in literature [10, 11], LocBLE employs a novel path loss exponent estimation based on data fusion of RSS readings and motion sensors. It enables light-weight path loss estimation on mobile devices.

9 DISCUSSION AND FUTURE WORK

We discuss LocBLE's usability and it's limitations in real-life scenarios. Finally, we showcase LocBLE's extensibility in future works.

9.1 On Accuracy of LocBLE

The design of LocBLE strives to achieve good accuracy by mitigating RSS signal fluctuations. The accuracy is around 1.8m in an indoor area, which is less accurate than other dedicated localization systems (*e.g.,* WiFi AP with a big antenna array). However, the flexibility (using commodity BLE beacon) and light-weighted (requiring only smartphone to operate) features together make LocBLE a unique and effective choice in the IoT paradigm (*e.g.,* help users find a lost item faster).

9.2 Limitations

Last meter navigation. LocBLE can navigate user to the target within 2m even from more than 10m away. However, it is not easy for LocBLE to make further improvement due to the nature of RSS. From our experiments, we observed that the Bluetooth proximity actually demonstrates fairly good accuracy within 2m. Therefore, if we incorporate proximity in LocBLE, we will be able to bring accuracy under 1m or even cm level. We leave this as our future work.

L-shaped measurement. LocBLE requires the observer to move "L" shape in the measurement stage. Although LocBLE only requires several steps of walk, it may still be inconvenient in some space limited area. To solve this difficulty, the observer may just walk straight and leave the symmetry problem to the navigation stage. During the last turn in navigation, we will know whether the observer is in a correct direction and correct him accordingly. Such design and implementation will be our future work.

Evaluation in crowded environments. Although LocBLE has shown to yield promising result given interference from WiFi in the vicinity, more experiments should be conducted in an environment with saturated interferences. For example, in a shopping mall where pedestrians' BLE signals and the surrounding BLE beacons create interferences and affect RSS readings. This scenario may require LocBLE to tune location estimation based on channel interference.

9.3 Extensibility of LocBLE

3-dimensional measurement. The current LocBLE is designed to show beacons' locations in a 2-D space. However, in real-world scenarios, users may prefer a 3-D location information for a more engaging experience (e.g., locating tagged devices more accurately on the user's AR device). 3-D localization can be done by modifying our data fusion and L-shaped movement. We leave the detailed design and evaluation of this as our future work.

Compatibility with Bluetooth 5.0. Bluetooth 5.0, as the next standard of BLE technology, offers four major improvements, including wider coverage [48]. This feature will enhance LocBLE's performance while keeping it still compatible with the upcoming Bluetooth 5.0.

10 CONCLUSION

We have presented LocBLE, a novel framework for locating BLE beacons on smartphones. It is shown to be able to achieve m-level accuracy with only commodity smartphone and proximity BLE beacons. By recognizing the environmental changes and multi-nodes scenarios, LocBLE can significantly enhance its location estimation accuracy. It provides a practical location estimation method for low-power IoT devices, enabling many IoT apps that have not been possible before.

REFERENCES

[1] How beacons will influence billions in us retail sales. http://www.businessinsider.com/beacons-impact-billions-in-reail-sales-2015-2.

[2] Health media network announces partnership with gimbal, inc. https://www.gimbal.com/press-releases/health-media-network-announces-partnership-with-gimbal-inc-creating-the-first-and-largest-healthcare-proximity-beacon-network-in-the-u-s/.

[3] Waze launches bluetooth beacons to avoid tunnel blackouts. https://techcrunch.com/2016/09/21/waze-launches-bluetooth-beacons-to-avoid-tunnel-blackouts/.

[4] How beacons can reshape retail marketing. https://www.thinkwithgoogle.com/articles/retail-marketing-beacon-technology.html.

[5] Souvik Sen, Jeongkeun Lee, Kyu-Han Kim, and Paul Congdon. Avoiding Multipath to Revive Inbuilding WiFi Localization. In *ACM MobiSys*, 2013.

[6] Ramsey Faragher and Robert Harle. An analysis of the accuracy of bluetooth low energy for indoor positioning applications. In *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014), Tampa, FL, USA*, volume 812, page 2, 2014.

[7] W. W. L. Li, R. A. Iltis, and M. Z. Win. A smartphone localization algorithm using rssi and inertial sensor measurement fusion. In *IEEE Global Communications Conference (GLOBECOM)*, pages 3335–3340, Dec 2013.

[8] Gaddi Blumrosen, Bracha Hod, Tal Anker, Danny Dolev, and Boris Rubinsky. Enhanced calibration technique for rssi-based ranging in body area networks. *Ad Hoc Netw.*, 11(1):555–569, January 2013.

[9] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge university press, 2005.

[10] Guoqiang Mao, Brian D.O. Anderson, and BarÄśÅ§ Fidan. Path loss exponent estimation for wireless sensor network localization. *Computer Networks*, 51(10):2467 – 2483, 2007.

[11] J. Koo and H. Cha. Localizing wifi access points using signal strength. *IEEE Communications Letters*, 15(2):187–189, February 2011.

[12] Jue Wang and Dina Katabi. Dude, where's my card?: Rfid positioning that works with multipath and non-line of sight. In *ACM SIGCOMM*, 2013.

[13] Longfei Shangguan, Zheng Yang, Alex X. Liu, Zimu Zhou, and Yunhao Liu. Relative localization of rfid tags using spatial-temporal phase profiling. In *USENIX NSDI*, 2015.

[14] NFC White Paper. https://www.ecma-international.org/activities/Communications/tc32-tg19-2005-012.pdf.

[15] Title 47 cfr part 15. http://www.ecfr.gov/cgi-bin/47cfrv1_02.tpl.

[16] Bluetooth 5.0 core specification, transmitter characteristics, page 2536. https://www.bluetooth.com/specifications/adopted-specifications.
[17] BLE Specification. https://www.bluetooth.com/specifications.
[18] Bluetooth low energy beacons. http://www.ti.com/lit/an/swra475a/swra475a.pdf.
[19] Bcm4339 datasheet. http://www.cypress.com/file/298016/.
[20] SVM in sklearn module. http://scikit-learn.org/stable/modules/svm.html.
[21] Anonymity. Locating and Tracking BLE Beacons with Smartphones. Technical report, Please refer to PC Chair for a Technical Report copy.
[22] Dongyao Chen, Kyong-Tak Cho, Sihui Han, Zhizhuo Jin, and Kang G. Shin. Invisible sensing of vehicle steering with smartphones. In *ACM MobiSys*, 2015.
[23] Alex T. Mariakakis, Souvik Sen, Jeongkeun Lee, and Kyu-Han Kim. Sail: Single access point-based indoor localization. In *ACM MobiSys*, 2014.
[24] Nirupam Roy, He Wang, and Romit Roy Choudhury. I am a smartphone and i can tell my user's walking direction. In *ACM MobiSys*, 2014.
[25] Yuanchao Shu, Kang G Shin, Tian He, and Jiming Chen. Last-mile navigation using smartphones. In *ACM MobiCom*, 2015.
[26] Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, and Feng Zhao. A reliable and accurate indoor localization method using phone inertial sensors. In *ACM UbiComp*, 2012.
[27] Donald J. Bemdt and James Clifford. Using dynamic time warping to find patterns in time series, 1994.
[28] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data*, 2004.
[29] swix. http://stsievert.com/swix/.
[30] Mohamed Boucadair, Reinaldo Penno, and Dan Wing. Universal plug and play (upnp) internet gateway device-port control protocol interworking function (igd-pcp iwf). 2013.
[31] Azkario Rizky Pratama, Risanuri Hidayat, et al. Smartphone-based pedestrian dead reckoning as an indoor positioning system. In *System Engineering and Technology (ICSET), 2012 International Conference on*, pages 1–6. IEEE, 2012.
[32] Estimote Beacons. http://estimote.com/.
[33] Radbeacon usb dongle. http://www.radiusnetworks.com/.
[34] LocBLE Demo. https://goo.gl/6MJOeX.
[35] Dartle iBeacon Locator. https://cloud.dartle.io/.
[36] XCode IDE for iOS development. https://developer.apple.com/xcode/ide/.
[37] Apple iBeacon. https://developer.apple.com/ibeacon/, 2013.
[38] Eddystone Beacon. https://developers.google.com/beacons/.
[39] Altbeacon. http://altbeacon.org/, 2015.
[40] Jie Xiong and Kyle Jamieson. ArrayTrack: A Fine-grained Indoor Location System. In *USENIX NSDI*, 2013.
[41] Swarun Kumar, Ezzeldin Hamed, Dina Katabi, and Li Erran Li. LTE Radio Analytics Made Easy and Accessible. In *ACM SIGCOMM*, 2014.
[42] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. SpotFi: Decimeter Level Localization Using WiFi. In *ACM SIGCOMM*, 2015.
[43] P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *IEEE INFOCOM*, 2000.
[44] Moustafa Youssef and Ashok Agrawala. The horus wlan location determination system. In *ACM MobiSys*, 2005.
[45] Zengbin Zhang, Xia Zhou, Weile Zhang, Yuanyang Zhang, Gang Wang, Ben Y. Zhao, and Haitao Zheng. I am the antenna: Accurate outdoor ap location using smartphones. In *ACM MobiCom*, 2011.
[46] Hyojeong Shin, Yohan Chon, Kwanghyo Park, and Hojung Cha. Findingmimo: Tracing a missing mobile phone using daily observations. In *ACM MobiSys*, 2011.
[47] E. Au. The latest progress on ieee 802.11mc and ieee 802.11ai [standards]. *IEEE Vehicular Technology Magazine*, 11(3):19–21, Sept 2016.
[48] Bluetooth 5.0 standard. https://www.bluetooth.com/specifications/adopted-specifications.