GauPro: An Accuracy-Improved Indoor Positioning System Based on Beacon Probabilistic Fingerprint

Yu Zhang*†, Zhongzheng Lai*†, Dong Yuan*, Wei Bao*, Bing Bing Zhou*, Jing Qiu*, Shen Wang†, Stewart Adams†
*Faculty of Engineering, The University of Sydney, NSW 2006, Australia
†CHUR Networks, Merewether Building H04, Darlington, NSW 2006, Australia
{yzha0058,zlai0652}@uni.sydney.edu.au, {dong.yuan,wei.bao,bing.zhou,jeremy.qiu}@sydney.edu.au, {david,stewart}@churinc.com

***Abstract*—With the increasing demand for indoor location based services and the fast growth of related technologies, indoor positioning services are becoming more and more popular and important. Multiple approaches have been studied in this field for the past two decades, but the performance is still not as satisfying as the outdoor positioning services. In this research, we study several mainstream indoor localisation technologies and propose GauPro, which is a Beacon probabilistic fingerprint algorithm that utilises the Gaussian Kernel model to improve the accuracy of traditional algorithm. Based on GauPro, we design and implement an indoor positioning system that can be easily deployed without any hardware modification of the receiving-end devices thereby providing positioning services for general smartphone users. For the evaluation, we deploy the system in a work space and set up two different field experiment scenarios. The first scenario is a rooms and corridors area and the second one is an empty hall area. The experimental results show that our GauPro algorithm-based system outperforms the traditional Bayesian probability-based approach by 17.3% in terms of error distance.**

I. INTRODUCTION

The Location Based Service has become more and more popular in recent years. The aim of such service is to provide detailed location information to customers and help them under different scenarios. For example, help guiding people to their destination in a city or finding the right place they need. The Global Positioning System (GPS) technology has made outdoor positioning highly successful and reliable. However, due to the high interference of the GPS signal inside a building, the indoor GPS accuracy is not as satisfying as the outdoor GPS.

Many systems have been built to solve the indoor positioning problem with various technologies, including and not limited to infrastructure-based [1], RF-based [2][3], magnetic-based [4] and camera-based [5]. However, many solutions involve modifications of the user's receiving-end device or require the user to carry a specific tag to assist the positioning process, which brings difficulties for their promotion in industry and brings troubles for the user.

Meanwhile, with the fast-growing of smart mobile phone and the ubiquitous computing technology, it has become easier for us to build a positioning system to track the mobile phone inside a building without too many other devices. Among all the ubiquitous technologies for indoor positioning, the Bluetooth-based approach is one of the most widely used technologies. It works similar to Wi-Fi but provides better accuracy since it covers less range and consumes less power. Two main directions for the Bluetooth-based approaches can be found in previous research. The first one is the triangulation method which studies on the relationship between signal strength and the distance of Bluetooth signal [6]. Another direction for Bluetooth positioning is the fingerprint method. A fingerprint, which is usually consists of several signal sources' information, represents a position on the map. It can be a piece of single dimension or multi-dimension information. People use the signal data that they receive in real-time to match the fingerprint stored in the database and locate the receiving device on the map. In this paper, we mainly discuss the fingerprint approach.

In the fingerprint approach, checkpoints are the selected locations in the indoor space to record the fingerprints. Traditionally, the Bayesian [7] is used to recognizes the prior probability of each checkpoint equally. In practice, however, the positioning result from the last timestamp should have a substantial impact on current estimation. In this paper, we try to use the Gaussian Kernel to control the prior probability of checkpoints. Gaussian Kernel is often used in the process of machine learning, specifically their support-vector machines (SVMs), as an algorithm for pattern analysis to find and study general types of relations. We take the advantage of it in this paper to calculate the prior probabilities of all checkpoints and develop an accuracy-improved indoor positioning system for general smartphone users. The key contributions of our work are as follows:

- We design GauPro, a Gaussian Kernel-based Bayesian algorithm. In this algorithm, the prior probabilities of each checkpoint are not equal, instead they will change according to the user's current position and moving direction.
- We implement a real indoor positioning system based on the GauPro algorithm including the client application and the back-end server. The system can be easily deployed in any indoor space without any hardware modifications of receiving-end devices, hence it can provide services for general smartphone users.
- We deploy the implemented system in a work space and conduct field experiments in difference scenarios. Using the real experimental data obtained in the field experiments, we perform evaluations of our GauPro algorithm-

based indoor positioning system.

The remainder of the paper is organised as follows. Section II introduces the overall architecture of the system and the design of our GauPro algorithm. Section III presents the implementation of the indoor localisation system, section IV evaluates the proposed system through field experiments. Section V discusses the related work to this paper and section VI draws the conclusion.

II. SYSTEM DESIGN AND THE GAUPRO ALGORITHM

A. *System Overview*

The purpose of our system is locating users while they are inside a building using fingerprint technology. We designed and implemented our system which provides indoor positioning service based on Beacon device, user's smartphone and several back-end servers. The phone application was developed on iOS system but the same Beacon protocol can also be used on Android.

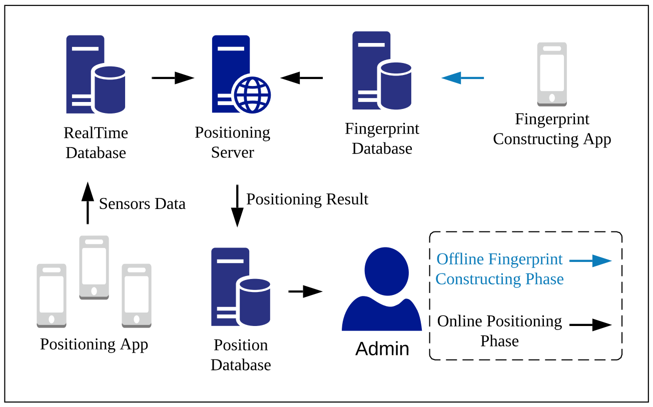

Fig. 1. Overview of the indoor positioning system

Our system consists of two main phases: **offline fingerprint database constructing** and **online positioning**. During the offline (fingerprint constructing) phase, we set checkpoint on the map and record the observed Bluetooth Received Signal Strength (RSS) value and store them in a static database. For the online (positioning) phase, we use a matching algorithm to match the observed run-time value with the values we stored in database.

Figure 1 shows an overview of our Beacon positioning system. We have the following components:

- **Positioning App**, smartphone App for user. Receive Bluetooth signals and send them to the Real-time database
- **Fingerprint constructing App**, smartphone App for admin. Collect RSS fingerprint and upload them to the fingerprint database
- **Real-time database**, temporarily stores RSS data uploaded by users to avoid data traffic problem.
- **Fingerprint database**, stores fingerprint information for all the checkpoints.
- **Positioning server**, estimate user's position with our designed algorithm.
- **Position database**, stores the calculated position information of each user.
- **Admin** can read all the user position information from the position database.

For the offline fingerprint constructing phase, we use a fingerprint collecting application to upload the gathered Received Signal Strength Indicator (RSSI) as fingerprint for each checkpoint. For the online positioning phase, the user's smartphone can receive wireless signals from several Beacon devices inside the building. The smartphone sends all the data to a real-time database regularly to avoid data traffic issue. Then, the core back-end server (Positioning Server) will cache data from both the real-time database and the fingerprint database, estimate the user's location and then send it to system admin or back to the user.

B. *Offline Phase*

The RSSI values (in dBm) observed from a checkpoint are subject to Gaussian Distribution

$$X \sim \mathcal{N}(\mu, \sigma^2). \tag{1}$$

The fingerprint (from one single AP) of a checkpoint i can be described as

$$F_{ij} \triangleq (CP_i, AP_j, \mu_{ij}, \sigma_{ij}), \forall i \in 1, ..., N, \forall j \in 1, ..., n, \tag{2}$$

where N is the total number of checkpoints, n represents the number of Beacons a checkpoint can hear. CP_i represents the checkpoint ID, AP_j is the AP's ID, μ_{ij} represents the mean RSSI value, and σ_{ij} represents the standard deviation of that AP. For each checkpoint i, we define their coordinate as (x_i, y_i). All the values in the fingerprint are measured and stored in the database in the offline phase.

We use the joint probability distribution function of a checkpoint's RSSI to represent the fingerprint. In our design, we consider the RSSIs collected from different access point checkpoints are independent to each other. We can hear n Beacons at checkpoint i. The fingerprint at checkpoint i can be considered as

$$f_i \triangleq [F_{i1}, F_{i2}, F_{i3}, ..., F_{in}].$$

We build the fingerprint database with fingerprint constructing App and a uploading script. After we have built a radio map and deployed the Beacon devices inside the area, we manually set checkpoints on the map. Then, we hold our phone (installed the fingerprint constructing App) and record the RSSI values of each checkpoint. Finally, we upload all the recorded fingerprint to a fingerprint database via a uploading script written in Python.

Then, we are going to use this database for our position estimation in the online phase.

C. Online Phase

1) Traditional Bayesian algorithm: When we collect signal with a mobile phone at checkpoint i, we will receive multiple RSSIs from different APs. For a single access point j, the received RSSI R_{ij} is subject to Gaussian distribution. Let the probability density function $g(R_{ij})$ as

$$g(R_{ij}) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp\left\{\frac{-(R_{ij}-\mu_{ij})^2}{2\sigma_{ij}^2}\right\}. \tag{3}$$

The reading of signal strength is discrete (integer). We normalize the probability distribution. During our test, we receive the RSSI between -50dBm and -90dBm. Thus, the signal strength is in [-50, -90]. Any reading we receive is an integer in this range. Thus, the normalized probability of having a specific reading $\hat{R_{ij}}$ at checkpoint CP_i is:

$$P(\hat{R_{ij}}|CP_i) = \frac{g(\hat{R_{ij}})}{\sum_{k=-50}^{-90} g(k)}. \tag{4}$$

The number of Beacons we can hear is n. Thus, the reading set can be expressed as $\hat{r_i}$ = [$\hat{R_{i1}}$, $\hat{R_{i2}}$, $\hat{R_{i3}}$, ..., $\hat{R_{in}}$]. Thus, if we want to estimate the joint probability distribution $P(\hat{r_i}|CP_i)$, we can now estimate the product of all the marginal probability distribution $P(\hat{R_{ij}}|CP_i)$. Combining equation (4), we have the probability of having a set of reading $\hat{r_i}$ at checkpoint i:

$$P(\hat{r_i}|CP_i) = \prod_{j=1}^{n} P(R_{ij}|CP_i). \tag{5}$$

To calculate the probability for each checkpoint given a specific reading, we can use Bayesian formula

$$P(CP_i|\hat{r_i}) = \frac{P(\hat{r_i}|CP_i)P(CP_i)}{\sum_{i=1}^{N} P(\hat{r_i}|CP_i)P(CP_i)}. \tag{6}$$

In equation (6), $P(CP_i)$ is the prior probability. In the traditional Bayesian algorithm, this is uniformly distributed. After we have calculated $P(CP_i|\hat{r_i})$ for all checkpoints, we can get a list of results.

The resultant location is averaged among the top three checkpoints (with the largest $P(CP_i|\hat{r_i})$ values). $L^* = (x^*, y^*)$:

$$x^* := \sqrt{X_1^2 + X_2^2 + X_3^2}, \tag{7}$$

$$y^* := \sqrt{Y_1^2 + Y_2^2 + Y_3^2}, \tag{8}$$

where X_1, X_2, X_3 and Y_1, Y_2, Y_3 are the coordinates of the top three checkpoints.

2) Improvement with Gaussian kernel: We notice that the prior probability of each checkpoint could be improved. Since the positioning is a dynamic process, the previous location can help locate the current one. Suppose the user's last known position is position $L^- = (x^-, y^-)$, and Δt time has passed. L^- is the outcome of L^* in the previous round. We also maintain $(q_0^*, q_1^*, q_2^*, q_3^*)$ to represent the current orientation vector, and θ^* to represent the current orientation angle. According to the differential equation for quaternion in [8], we can use our readings from the gyroscope $\hat{g_x}$, $\hat{g_y}$, $\hat{g_z}$, which are the angular velocity on each axis, and our time shift Δt to update the quaternion:

$$\begin{bmatrix} q_0^* \\ q_1^* \\ q_2^* \\ q_3^* \end{bmatrix} := \begin{bmatrix} q_0^- \\ q_1^- \\ q_2^- \\ q_3^- \end{bmatrix} + \frac{1}{2}\cdot\Delta t\cdot \begin{bmatrix} -\hat{g_x}\cdot q_1 - \hat{g_y}\cdot q_2 - \hat{g_z}\cdot q_3 \\ \hat{g_x}\cdot q_0 - \hat{g_y}\cdot q_3 + \hat{g_z}\cdot q_2 \\ \hat{g_x}\cdot q_3 + \hat{g_y}\cdot q_0 - \hat{g_z}\cdot q_1 \\ -\hat{g_x}\cdot q_2 + \hat{g_y}\cdot q_1 + \hat{g_z}\cdot q_0 \end{bmatrix}, \tag{9}$$

$(q_0^-, q_1^-, q_2^-, q_3^-)$ are the values of $(q_0^*, q_1^*, q_2^*, q_3^*)$ in the last round. The initial values of $(q_0^-, q_1^-, q_2^-, q_3^-)$ are $(1, 0, 0, 0)$. Next, we derive the rotation angle $\Delta\theta$ around z axis with the updated $(q_0^*, q_1^*, q_2^*, q_3^*)$:

$$\Delta\theta := -\arctan\left(\frac{q_1^* q_2^* + q_0^* q_3^*}{q_0^{*2} - q_1^{*2} + q_2^{*2} - q_3^{*2}}\right), \tag{10}$$

$$\theta^* := \theta^- + \Delta\theta, \tag{11}$$

θ^- is the value of θ^* in the last round. The initial value of θ^- is 0.

Now that we have the rotation angle $\Delta\theta$, we need to know whether the user is moving. To simplify our calculation, we set a threshold on the phone's acceleration. Once we receive a reading higher than that threshold, we assume that the user has moved one step. If so, we can use the previous positioning result: $L^- = (x^-, y^-)$, last orientation angle θ and a step length l to calculate the new position $L^* = (x^*, y^*)$:

$$x^* := x^- + l\cdot\cos(\theta^*), \tag{12}$$

$$y^* := y^- + l\cdot\sin(\theta^*), \tag{13}$$

l is the default step size, which is given in advance. After we have successfully estimated position change in Δt, we can use the distance between them as our new step length.

We now introduce the method we use to reallocate the prior probability of each checkpoint in Section II-C1. In our design, we use Gaussian Kernel to achieve so.

Gaussian Kernel, or Radial basis function (RBF) kernel, is a useful kernel function that can map lower dimension points to higher dimension space. It describes a function of the squared Euclidean distance between two locations. Suppose we have a reference location (x', y') and a checkpoint at (x_i, y_i), we calculate the kernel function $k(x_i, x', y_i, y')$ as follows

$$k(x_i, x', y_i, y') \triangleq \exp\left\{-\frac{(x_i - x')^2 + (y_i - y')^2}{2\sigma^2}\right\}. \tag{14}$$

We use L^* as the reference point in equation (14), we can have the kernel value of that checkpoint regarding point (x^*, y^*). σ is the average step length we have measured so far. To simplify the following calculation, we define the checkpoint CP_i's kernel value as Z_i:

$$\begin{aligned} Z_i &:= k(x_i, x^*, y_i, y^*) \\ &= \exp\left\{-\frac{(x_i - x^*)^2 + (y_i - y^*)^2}{2\sigma^2}\right\}. \end{aligned}$$

Once we have finished calculating from Z_1 to Z_N, we normalise the result of each checkpoint to get their new prior probability $P^{'}(CP_i)$:

$$P^{'}(CP_i) := \frac{Z_i}{\sum_{i=1}^{N} Z_i}. \tag{15}$$

Finally, by combining equation (6) and (15), our Bayesian formula $P(CP_i|\hat{r_i})$ now becomes $P^{'}(CP_i|\hat{r_i})$

$$P^{'}(CP_i|\hat{r_i}) = \frac{P(\hat{r_i}|CP_i)P^{'}(CP_i)}{\sum_{i=1}^{N} P(\hat{r_i}|CP_i)P'(CP_i)}. \tag{16}$$

The $P(\hat{r_i}|CP_i)$ remains the same as in equation (3) and (4).

Once we have finished computing the $P^{'}(CP_i|\hat{r_i})$ for every checkpoint, we can use equation (7)(8) again to calculate the resultant location $L^{*^{'}} = (x^{*^{'}}, y^{*^{'}})$, which we consider as the L^{-} in the next Δt time.

$$x^{*} := \sqrt{X_1'^2 + X_2'^2 + X_3'^2},$$

$$y^{*} := \sqrt{Y_1'^2 + Y_2'^2 + Y_3'^2},$$

where $X_1^{'}, X_2^{'}, X_3^{'}$ and $Y_1^{'}, Y_2^{'}, Y_3^{'}$ are the coordinates of the top three checkpoints in the result list we calculated with equation (16).

III. SYSTEM IMPLEMENTATION

In this section, we will talk about the implementation of the algorithm. We will start with the fingerprint database, then the mobile app we used to collect fingerprint. Next, we will introduce the server that runs our algorithm, including the data structure on back-end and the flow chart of our code.

A. Fingerprint Database

The fingerprint database we constructed consists of 3 key tables:

- **beacon** stores all the beacon devices deployed inside the area. Each is represented by a unique ID, a name, latitude and longitude. Although the algorithm does not need the position of beacon devices at all, we need this information to display it on the user interface.
- **checkpoint** contains the information of all checkpoints. The key elements in this table are checkpoint ID and the exact position on the map. The precision of the final result rely on the location of each checkpoint. So, it is vital to keep this information in the database, and not changing them once it has been recorded.
- **beacon_fingerprint** stores the fingerprint information. As we have described in section II, the fingerprint of a checkpoint consists of 4 elements: checkpoint's ID, AP's ID, average RSSI and RSSI's standard deviation.

B. Mobile App Implementation

1) App for fingerprint construction: This app is for service providers, e.g. the building manager, to collect the fingerprint and construct the database. Before collecting the fingerprint, we need to store the data for Beacon and checkpoints first. For table **beacon**, we store the Beacon devices' location and IDs. For table **checkpoint**, we manually add checkpoints with our online fingerprint constructing application which connects directly with the fingerprint database and provides a user interface to display the radio map.

As shown in Figure 2, the green dot represents the checkpoint, and the blue sign represents Beacon location. Once they are added to the map and saved, the information will be automatically stored into the corresponding table.

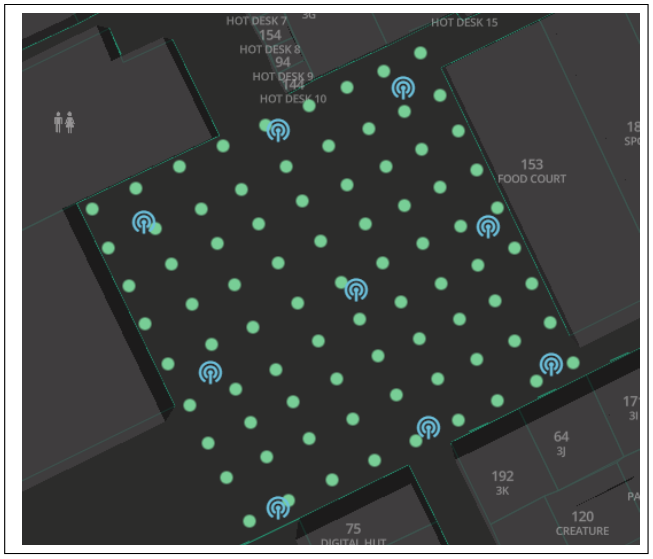

Fig. 2. Our online fingerprint constructing App and our Beacon/Checkpoints

The mobile application we used to collect fingerprint can record all the received RSSI from nearby beacon devices and identify their ID. The data can be harvested approximately every second and stored locally. The workflow of offline fingerprint construction is shown in Figure 3. As we can see from the figure, once our application is started, it begins to detect the nearby Beacon signals. For each checkpoint we store about 100 set of data for the following calculation. Once the data is collected, we run a python script to calculate their average value and standard deviation, form a fingerprint data structure and upload it to the fingerprint database.

2) App for positioning: This app is for general smartphone users to do indoor positioning. The app will scan the nearby Beacon signal and upload it to the realtime database. It will provide the raw data about our Bluetooth positioning. The information includes device serial number, timestamp, Beacon sniff, gyroscope and accelerometer data. For Beacon, we use the CoreLocation framework provided by Apple to fetch the Bluetooth signal and their sniff. For the gyroscope and accelerometer, we use the CoreMotion framework to collect

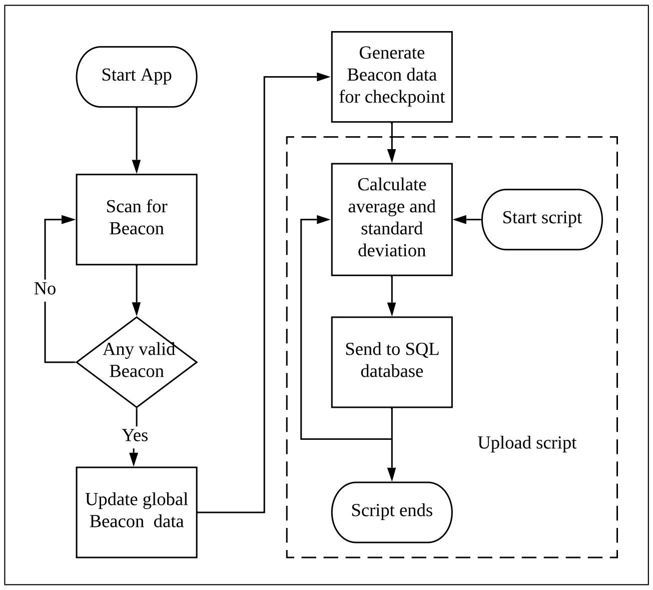

Fig. 3. Workflow of fingerprint construction

information from our phone's sensor[1].

The device serial number is used to identify our testing device, the timestamp records when we upload the data. Beacon sniff contains the detected Beacon's unique ID, signal strength, and the calculated distance. For the gyroscope and accelerometer, we send the raw data detected by the smartphone's sensor. The structure of uploaded data is defined as follows:

```
position
└─device serial number
   └─timestamp
      ├─sniff
      │  └─beacon
      │     └─UUID
      │        ├─Signal_strength
      │        └─distance
      ├─gyroscope
      │  └─g_x, g_y, g_z
      └─acclerometer
         └─a_x, a_y, a_z
```

As we have described in Section II-C2, we can use g_x, g_y, g_z to calculate the rotation angle with equation (7) (8), and use a_x, a_y, a_z to decide whether the user is moving by calculating the sum of these three value and compare it with our threshold.

C. Back-End Server Implementation

The back-end server is where we run our algorithm and find out the positioning result. The server is written in Golang because it is open-source, fast and concurrent, which is essential for a back-end server. Being open-source means we have a lot of documents and help threads online to read. Being fast and concurrent means our service will efficiently run since we expect more than five thousand people to use the system at the same time ideally. In this section, we will explain the structure and workflow in details.

[1]Our current implementation is based on the iOS system for iPhone. For Android smartphones, these sensor data can be obtained and processed similarly, please refer to https://developer.android.com/guide/topics/sensors/sensors_overview.

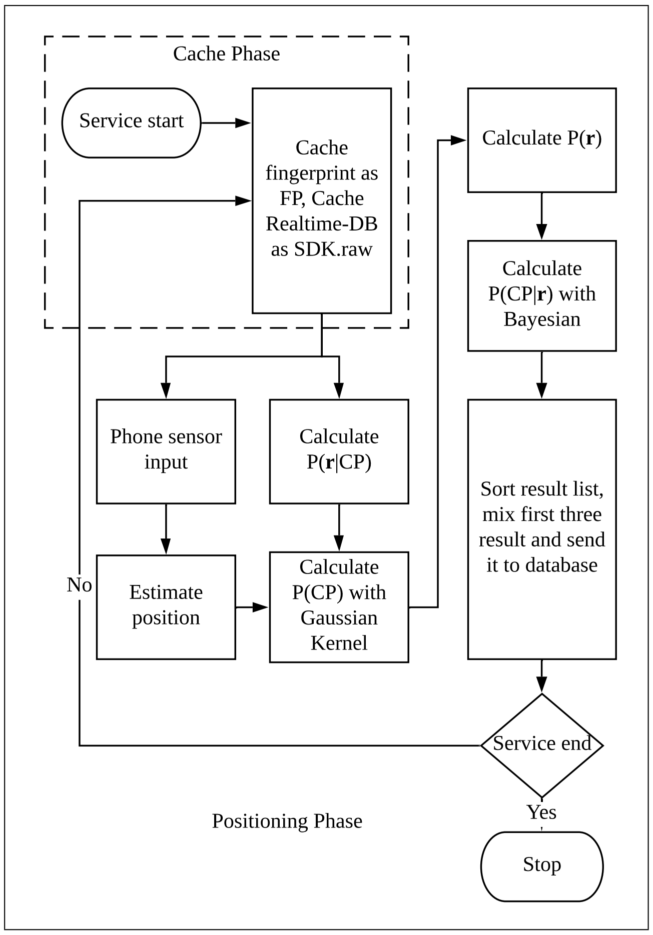
Cache Phase
Service start
Cache fingerprint as FP, Cache Realtime-DB as SDK.raw
Calculate P(**r**)
Calculate P(CP|**r**) with Bayesian
Phone sensor input
Calculate P(**r**|CP)
Sort result list, mix first three result and send it to database
No
Estimate position
Calculate P(CP) with Gaussian Kernel
Service end
Yes
Positioning Phase
Stop

Fig. 4. Workflow of back-end server

As shown in Figure 4, overall we have two phases: the cache phase and the positioning phase. The figure can be explained as follows: once we start our service, we first cache the Beacon sniff data and fingerprint we need; Then, we use the phone sensor data to estimate position, and calculate the components for our Bayesian formula; Finally, we calculate the result with Bayesian formula, sort the result list and get the final result.

For the cache phase, we need to load three tables in total and build corresponding data storage structure. We mainly need two parts of data:

- **Beacon sniff** received from realtime database
- **Beacon device** location and **checkpoint** information

So, we define data structure accordingly:

- **sdkRaw** - This data structure is to represent the real-time Beacon sniff uploaded by user. The detailed structure is as follows:

  `map[string]map[int64]sdk.Position`

  This is a key value structure. The first two map structure is the key, which stores user serial number and the timestamp. The value is the Beacon sniff, user's orientation and acceleration.

- **CheckpointFP** - This data structure stores the fingerprint information. Here is the detailed structure:

  ```
  map[int64]map[string]Sniff
  ```

  Similar to we have described above, the first two map structure is the key, represents checkpoint's ID and the Beacon's ID. The value is the sniff data.

We use GORM[2], which is a Object-relational Mapping (ORM) library for for dealing with relational databases, to connect the SQL database and cache their data. **sdkRaw** stores the observed Beacon sniff, **CheckpointFP** stores all the checkpoints' location and their average RSSI & standard deviation from each Beacon. Once we have all the cached data, we can proceed to the next phase.

To calculate the probability list for all checkpoints, we need three key components according to the Bayesian formula discussed in Section II:

- $P(\hat{r_i}|CP_i)$ - Calculate with Gaussian formula;
- $P(CP_i)$ - Value assigned according to Gaussian Kernel;
- $P(\hat{r_i})$ - The sum of all $p(\hat{r_i}|CP_i)$;

The result list is a PairList (a structure in Golang) which contains a list of checkpoint ID and their probability. As described in Section II, we sort the list by the probability values and choose the first three checkpoints, average their coordinates as our final result.

IV. Evaluation

In this section, we will evaluate the performance of the algorithm we proposed and discuss the obtained result. We will first introduce the experimental setup and the testing criteria for the indoor positioning system. Then we will run our system under different scenarios and analyze the result.

A. Experimental Setup

We deploy our system in a real indoor work space to conduct field experiment. The building we choose is 100 Harris St Sydney[3] that has a complex indoor structure. Inside the building, not only can we find areas with multiple rooms and corridors but also big empty halls as well. We deploy our Beacon devices on either the pillar or the ventilation duct to gain the best signal radiation. The first area we choose consists of 12 small rooms and a narrow corridor, with 6 Beacon APs deployed. We use this area to represent places like the workplace or school. The second one is a large empty hall with 9 Beacon APs deployed. We use it to represent areas like warehouses or train stations.

From Figure 5, we have a visual understanding of the testing areas. For the two testing areas we choose, they both have chairs, tables, plants and other office supplies. A few Wi-Fi access points are deployed inside the whole area as well as Bluetooth access points so that other Wi-Fi or Bluetooth signal can interfere with the signal from our testing device, which makes the testing area closer to a real-life scenario where the area is always full of noise and obstacles.

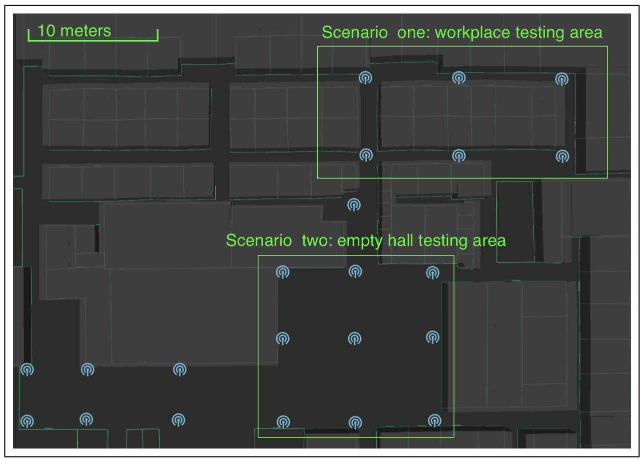
10 meters
Scenario one: workplace testing area
Scenario two: empty hall testing area

Fig. 5. Testing areas

B. Benchmark Approach and Testing Criteria

The benchmark approach we select for our algorithm is the traditional Bayesian method. It is a probabilistic algorithm that was initially proposed by Youssef et al. in [7]. The basic idea of the algorithm is to construct a radio map by collecting RSS, using the joint probability distribution as the checkpoint's fingerprint and use Bayesian algorithm to estimate the location of the user. In our GauPro algorithm, we use Gaussian Kernel to modify the prior probability in the Bayesian formula and implemented our own system for testing.

To evaluate an indoor positioning system based on Bluetooth technology, the most important properties of the system are error distance and precision:

- **Error distance** means the distance between our estimated position and actual position. Before we run the test, at each location, we keep a record of the actual position. When we finish our positioning process, the algorithm will return an estimated position, i.e., a pair of latitude and longitude. We then compare these two positions and calculate the distance between them, in meters.
- **Precision** measures the reliability and consistency of our algorithm. We can obtain precision by repeating multiple measurements under a persistent condition, and see how many times the result has appeared. For example, during 100 times of experiments at one location, 50% of the time a result of less than 3m has appeared, 80% of the time a result of less than 5m has appeared and 95% of the time a result of less than 10m has appeared.

To calculate the error distance and precision of our algorithm, we choose a start position and walk along a path consists of testing positions. We do repeat the experiment for multiple times to make the result more reliable. As for the default orientation angle in Section II-C2, we set the east direction in Figure 5 as the angle of 0. Our algorithm will return a result by its latitude and longitude so that we can calculate the distance between the result and the testing points.

[2]https://gorm.io/

[3]https://goo.gl/maps/1ceLQbx24BZE7mFRA

C. Positioning Result

We did separate experiments for both testing areas to ensure the performance of the system under different scenarios. We use tables and diagrams to analyze the result and compare our result with the traditional Bayesian positioning method to see if there is an improvement regrading the error distance and precision.

1) Scenario One - Empty Hall Area: This area has no wall and only contains some pillar, sofa and tables. We performed 12 times of positioning inside this area. The average accuracy in total is 2.543m, and the average standard deviation is 1.359. In order to make it easier for us to comparing the results from both algorithms, we generate a chart (Figure 6).

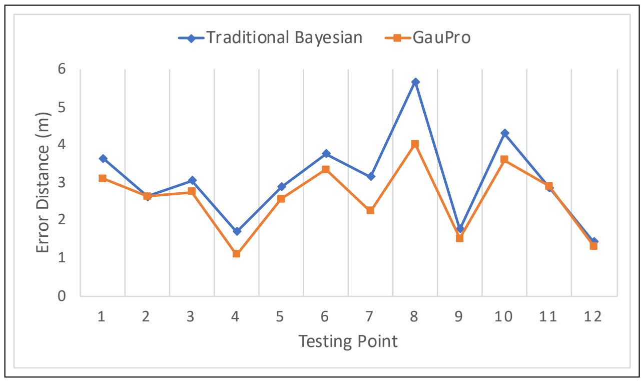

Fig. 6. Error distance comparison for empty hall area

We can tell from the chart that when implementing the improved algorithm, 10 out of 12 testing points can provide better error distance. Compared to the traditional algorithm, our error distance has improved by 17.3%. We discover that the improved algorithm can help to eliminate some of the results with big error distance. In other words, it can improve the worst scenarios of the original Bayesian algorithm. When we calculate position with the original algorithm, every checkpoint in the testing area has the same prior probability, which makes the positioning results evenly distributed around the testing point. However, when we estimate a position and use the Gaussian Kernel to assign the prior probabilities, checkpoints close to the estimated point will have higher priority. Thus, the checkpoints far from the estimated position are not likely to be chosen by the algorithm. Hence we can ignore the result that is not in the direction of the user's movement and usually has significant error distance.

We have another chart (Figure 7) to show precision differences more directly. The improved algorithm can provide a better result in all three ranges because the results are more likely to gather around the predicted point instead of spreading across the map. Therefore it is easier for our algorithm to get more consistent positioning results with higher accuracy.

2) Scenario Two - Workplace Area: Similar tests have also been performed to the workplace testing area. Inside this area, we have several walls made of glass and one pillar in each big room. We also have tables and chairs for people to use since

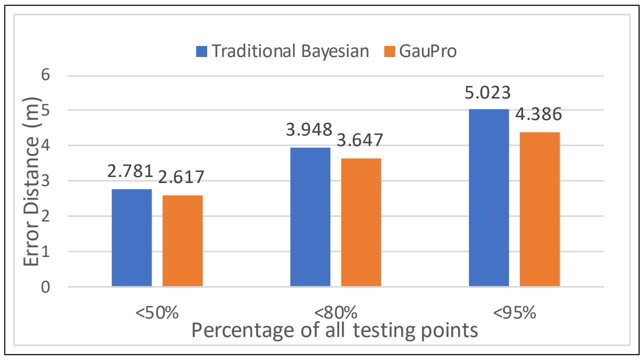

Fig. 7. Precision comparison for empty hall area

it is a real work space. The average accuracy for this testing area is 2.712m, and the standard deviation is 1.125.

We run the same tests with the original Bayesian algorithm. As shown in Figure 8, our algorithm has slightly improved 9 out of 12 results. Compared to the traditional algorithm, our error distance has improved by 8.1%. Figure 9 indicates that the precision for 50%, 80% and 95% of the time has been improved by 3.92%, 2.96% and 5.49% respectively.

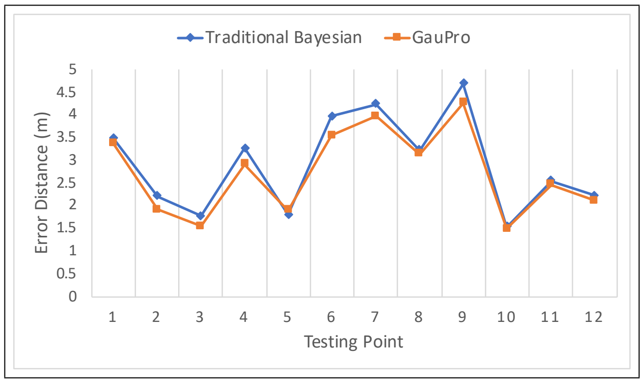

Fig. 8. Error distance comparison for workplace area

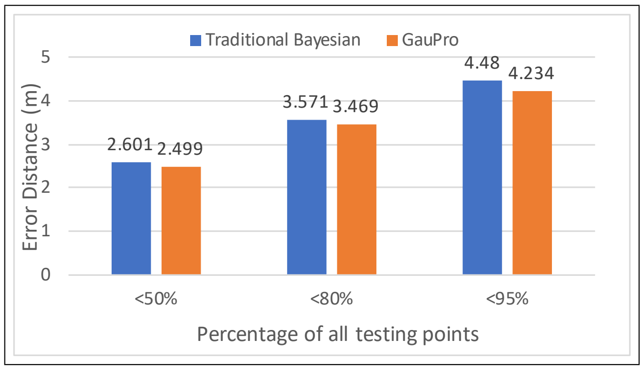

Fig. 9. Precision comparison for workplace area

V. Related Work

A. Overview of indoor positioning

A survey [9] has discussed different technologies for indoor localization from hundreds of literature. In project RADAR [2], the authors use Viterbi-like Algorithm for continuous user tracking and use it to guess the user's location. In [10], the authors divide the entire area into grid cell of 1 m and estimate the probability of each cell by calculating the Euclidean distance between the set of signal strength values in a map cell and the current fingerprint. A system called MoteTrack [11] proposed a decentralized location estimation protocol that relies only on local data, local communication, and operational nodes. It allows the base station nodes themselves to perform location estimation, rather than relying on a central server.

For Wi-Fi approach, Chronos [3] measures sub-nanosecond Time of Flight (ToF) on commercial Wi-Fi radios. Another example is SpotFi [12], which provide a better Angle of Arrive (AoA) estimation and direct path filtering. For Bluetooth technology, a research [13] was carried out by using the triangulation method combined with least square estimation. Sudarshan in his work [14] has described a study about choosing APs' location inside a room. Their solution is to find a maximum-resolution sub-hypergraph of the indoor Bluetooth signal.

Other than Wi-Fi or Bluetooth, people also came up with the idea of fingerprint database or landmark-based localization rely on the Earth's magnetic field. A system with an enhanced particle filter [15] was proposed, which build a database of magnetic field landmark. Another solution proposed by previous researchers is trying to make use of the backscatter signals introduced in system like HitchHike [16] and WiTag [17].

B. Positioning techniques

1) Triangulation positioning method: Among all the techniques of Bluetooth indoor positioning, one of the easier ways is the triangle localisation method based on received signal strength. The theory is quite simple: calculate the distance between the receiving device and multiple Bluetooth signal source, then use geometry knowledge and related algorithms to solve the problem. This paper [6] has analysed several different ways of implementing indoor triangulation algorithm including the three distance-based algorithms, least Square Estimation, three-border and centroid Method. These algorithms are required when all the received signals are collected and calculated, they are not likely intersecting into one single point in 99% of the time.

One of the most widely used algorithms in wireless positioning system is Least Square Estimation. This method is about "estimating parameters by minimising the squared discrepancies between observed data, on the one hand, and their expected values on the other" [18]. When having multiple intersecting points, with the help of a proper characteristic equation and weighting matrix, they can find a regression area or spot that will represent the result position of the mobile.

Given the location APs, triangulation method can be used to determine the position of a mobile device inside a building. The results provided by [6] are less than 3.5 meters for 95% of the time. One of the advantages of the triangulation method is that it runs very fast and costs less computing resources. The disadvantage of this method is that it relies heavily on the location of the APs in the algorithm.

2) Fingerprint method: Another common solution for Bluetooth indoor positioning is fingerprint method. There are two different kinds of approaches: deterministic and probabilistic. For the deterministic method, a paper from Microsoft [19] described the relationship between the received signal and signal fingerprint and how to compare them. It is called k-NN method. Based on that, an algorithm called weighted k-NN algorithm [20] was also studied to improve the drawback of the original k-NN algorithm where all k neighbours have the same vote.

The first probabilistic wireless positioning method was proposed by Youssef et al. [7]. They use the joint probability distribution as the fingerprint and use Bayesian formula to estimate the position. In Swangmuang and Krishnamurthy's work [21], they introduced a new analytical model for estimating the probability distribution of indoor Wi-Fi fingerprint. Based on their work, our designed system further improved positioning accuracy by leveraging the users' movement information and by redistributing the probabilities of each checkpoint.

VI. Conclusions

In this paper,we present the design, implementation and evaluation of a Beacon-based probabilistic fingerprint indoor positioning system. We estimate the user's relative position with the information from gyroscope and accelerometer and use Gaussian kernel to improve the prior probability calculation in the traditional Bayesian method. We set the traditional Bayesian method as the benchmark. According to our obtained experiment results, our system has fairly good error distance with an average of 2.543 and 2.712 meters for the two testing areas, improved the traditional algorithm by 17.3% and 8.1%. Our results appear to be more reliable and consistent when compared to the original one. The experiments were done in a real workplace environment where we encounter various interference including Wi-Fi signal, other Bluetooth signals from people's phones and the signal fluctuation caused by multiple people walking inside the area at the same time.

References

[1] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.

[2] P Bahl, Vn Padmanabhan, and A Balachandran. Enhancements to the RADAR user location and tracking system. *Microsoft Research*, 2000.

[3] Deepak Vasisht, Swarun Kumar, and Dina Katabi. Decimeter-level localization with a single WiFi access point. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016*, 2016.

[4] Hendrik Hellmers, Abdelmoumen Norrdine, Jörg Blankenbach, and Andreas Eichhorn. An imu/magnetometer-based indoor positioning system using kalman filtering. In *International Conference on Indoor Positioning and Indoor Navigation*, pages 1–9. IEEE, 2013.

[5] Dominik Van Opdenbosch, Georg Schroth, Robert Huitl, Sebastian Hilsenbeck, Adrian Garcea, and Eckehard Steinbach. Camera-based indoor positioning using scalable streaming of compressed binary image signatures. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2804–2808. IEEE, 2014.

[6] Yapeng Wang, Xu Yang, Yutian Zhao, Yue Liu, and Laurie Cuthbert. Bluetooth positioning using RSSI and triangulation methods. In *2013 IEEE 10th Consumer Communications and Networking Conference, CCNC 2013*, 2013.

[7] Sam H. Noh Moustafa A. Youssef, Ashok Agrawala, A. Udaya Shankar. A probabilistic clustering-based indoor location determination system, 2002.

[8] Simone Sabatelli, Marco Galgani, Luca Fanucci, and Alessandro Rocchi. A double-stage kalman filter for orientation tracking with an integrated processor in 9-d imu. *IEEE Transactions on Instrumentation and Measurement*, 62(3):590–598, 2012.

[9] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. A Survey of Indoor Localization Systems and Technologies. *IEEE Communications Surveys and Tutorials*, 2019.

[10] Ramsey Faragher and Robert Harle. Location fingerprinting with bluetooth low energy beacons. *IEEE Journal on Selected Areas in Communications*, 2015.

[11] Konrad Lorincz and Matt Welsh. MoteTrack: A robust, decentralized approach to RF-based location tracking. In *Lecture Notes in Computer Science*, 2005.

[12] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. SpotFi: Decimeter level localization using WiFi. In *SIGCOMM 2015 - Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015.

[13] Silke Feldmann, Kyandoghere Kyamakya, Ana Zapater, and Zighuo Lue. An indoor Bluetooth-based positioning system: Concept, implementation and experimental evaluation. In *Proceedings of the International Conference on Wireless Networks*, 2003.

[14] Sudarshan S. Chawathe. Beacon placement for indoor localization using Bluetooth. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2008.

[15] Fang Zhao Jose Luis Carrera V., Zhongliang Zhao, Torsten Braun, Haiyong Luo. Discriminative Learning-based Smartphone Indoor Localization, 2018.

[16] Pengyu Zhang, Dinesh Bharadia, Kiran Joshi, and Sachin Katti. HitchHike: Practical backscatter using commodity WiFi. In *Proceedings of the 14th ACM Conference on Embedded Networked Sensor Systems, SenSys 2016*, 2016.

[17] Manikanta Kotaru, Pengyu Zhang, and Sachin Katti. Localizing low-power backscatter tags using commodity WiFi. In *CoNEXT 2017 - Proceedings of the 2017 13th International Conference on emerging Networking EXperiments and Technologies*, 2017.

[18] Sara A. van de Geer. Least Squares Estimation. In *Encyclopedia of Statistics in Behavioral Science*. 2005.

[19] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings - IEEE INFOCOM*, 2000.

[20] Anja Bekkelien, Michel Deriaz, and Stéphane Marchand-Maillet. Bluetooth indoor positioning. *Master's thesis, University of Geneva*, 2012.

[21] Nattapong Swangmuang and Prashant Krishnamurthy. Location fingerprint analyses toward efficient indoor positioning. In *6th Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2008*, 2008.