Wireless Netw
DOI 10.1007/s11276-016-1309-9

SALA: Smartphone-Assisted Localization Algorithm for Positioning Indoor IoT Devices

Jaehoon (Paul) Jeong[1] · Solchan Yeon[2] · Taemoon Kim[3] · Hyunsoo Lee[3] · Song Min Kim[4] · Sang-Chul Kim[2]

Abstract This paper proposes a Smartphone-Assisted Localization Algorithm (SALA) for the localization of Internet of Things (IoT) devices that are placed in indoor environments (e.g., smart home, smart office, smart mall, and smart factory). This SALA allows a smartphone to visually display the positions of IoT devices in indoor environments for the easy management of IoT devices, such as remote-control and monitoring. A smartphone plays a role of a mobile beacon that tracks its own position indoors by a sensor-fusion method with its motion sensors, such as accelerometer, gyroscope, and magnetometer. While moving around indoor, the smartphone periodically broadcasts short-distance beacon messages and collects the response messages from neighboring IoT devices. The response messages contains IoT device information. The smartphone stores the IoT device information in the response messages along with the message's signal strength and its position into a dedicated server (e.g., home gateway) for the localization. These stored trace data are processed offline through our localization algorithm along with a given indoor layout, such as apartment layout. Through simulations, it is shown that our SALA can effectively localize IoT devices in an apartment with position errors less than 20 cm in a realistic apartment setting.

Keywords Indoor · Localization · Algorithm · Smartphone · IoT · Device · Trajectory

✉ Jaehoon (Paul) Jeong
pauljeong@skku.edu

Solchan Yeon
green1343@kookmin.ac.kr

Taemoon Kim
ktm0850@skku.edu

Hyunsoo Lee
stanlee5@skku.edu

Song Min Kim
ksong@cs.umn.edu

Sang-Chul Kim
sckim7@kookmin.ac.kr

[1] Department of Interaction Science, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon, Gyeonggi-do 440-746, Republic of Korea

[2] School of Computer Science, Kookmin University, 77 Jeongneung-ro Seongbuk-gu, Seoul 136-702, The Republic of Korea

[3] Department of Software, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon, Gyeonggi-do 440-746, Republic of Korea

[4] Department of Computer Science, George Mason University, 4400 University Drive, Fairfax, VA 22030, USA

1 Introduction

Recently the research for smart home has been spotlighted to facilitate the comfortable life of residents in home [1]. This smart home is accommodating an increasing number of Internet of Things (IoT) devices, such as mobile smart devices, appliances, and sensors. At this smart home, users want to remote-control and monitor IoT devices, such as smart TV, air conditioner, lights, temperature controller, entrance door lock, rice cooker, refrigerator, and robot cleaner. This remote-control and monitoring is possible due to the following two trends. As the first trend, for the easy management of IoT devices, AllSeen Alliance is developing a system called AllJoyn [2], which makes the IoT devices to advertise and share their abilities with other IoT devices near them. As shown in Fig. 1a, the solution in

Published online: 10 June 2016

Allseen Alliance lists IoT devices at home without their location information. On the other hand, as shown in Fig. 1b, it will be more convenient for the management of IoT devices when the positions of the IoT devices at home are visually displayed in the smartphone or tablet of a home user. Also, as a mobile node, a robot cleaner keeps moving during the cleaning in the apartment. A smartphone can track it through our SALA, so the home user can easily locate it in the apartment. As the second trend, a smartphone has many motion sensors for indoor positioning, such as accelerometer, gyroscope, and magnetometer. The sensor-fusion of these sensors allows the smartphone to keep its mobility trajectory indoors, such as at home and in office [3, 10]. *A natural research question* is how to utilize the smartphone's positioning capability to localize IoT devices at home.

Let us provide a gap analysis between the legacy indoor localization research and our localization problem of indoor IoT devices. The legacy indoor localization research has been so far focused on measuring the position of a mobile device itself (e.g., smartphone) rather than the positions of its neighboring devices indoors (e.g., smart TV, air conditioner, and robot cleaner at smart home, as shown in Fig. 1b). There are three approaches of indoor localization, such as (1) WiFi fingerprint (using signal strength measurement for WiFi access points (APs)) [4–7], (2) Range-based localization (using distance measurement hardware) [8, 9], and (3) Dead reckoning (using motion sensors embedded within a smartphone) [3, 10, 13]. First, WiFi fingerprint is based on signal strength from nearby APs. Received Signal Strength (RSS) is proportional to the distance between a signal transmitter (e.g., AP) and a signal reciever (e.g., smartphone). From this fact, at a certain location, a list of the RSSes from multiple nearby APs can be used like a fingerprint. A manager system for constructing fingerprints in an indoor map (e.g., apartment and office) collects the lists of RSSes at the intended positions as landmarks. It offers them to user devices (e.g., smartphones) so that they can know their own positions by comparing the pairs of the RSS list and the corresponding position, which are provided by the manager system. Second, to overcome the unstableness of WiFi fingerprint over time, Range-based localization uses ranging hardware for distance measurement, such as acoustic signal beacons. After the distance measurements through the accurate ranging between a smartphone and each anchor, the smartphone can be accurately localized by multilateration. Third, Dead reckoning is based on the motion sensors of a smartphone, such as accelerometer, gyroscope, and magnetometer. These motion sensors measure the physical acceleration, angular velocity, and magnetic force (i.e., the direction and strength of magnetic field). The user devices collect these sensing data and update their physical positions along their movement over time. In the environments where a small number of APs are located, the dead reckoning shows better accuracy than the WiFi fingerprint; note that a small number of APs may make a weak signature of WiFi fingerprint, and the dead reckoning is less affected by the number of APs. Also, the range-based localization requires additional hardware for ranging, so it is a costly approach for the localization of IoT devices. Note that the dead reckoning can be applied for the localization of a smartphone rather than its neighboring IoT devices indoors. Thus, the research on localization of the IoT devices has been insufficiently conducted. In our study, we use a smartphone as a mobile anchor to localize these IoT devices indoors.

This paper proposes a Smartphone-Assisted Localization Algorithm (called SALA) that is the first work for the localization of IoT devices for the visual display of IoT devices' positions through the smartphone indoor positioning. When a smartphone enters its home, it keeps

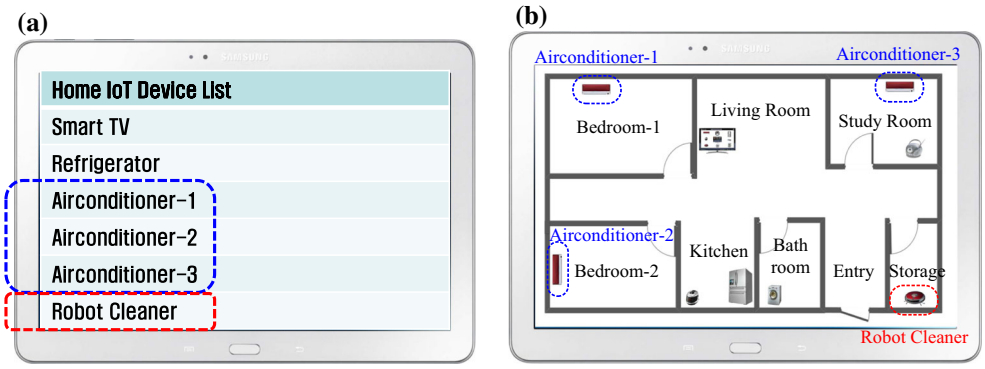


Fig. 1 Display methods of indoor IoT devices at smart home. **a** Text-based display by AllJoyn shows only the list of devices, so it is hard to distinguish three airconditioners. **b** Graphic-based display by SALA shows the physical positions of the devices, so it is easy to distinguish three airconditioners

tracking its indoor position by the functionality of self-tracking [12, 13]. Since the smartphone has motion sensors, it can track its position during its movement without GPS signal in indoor environments. The smartphone plays a role of a mobile beacon while moving around the home. It periodically broadcasts short-distance beacon messages. When an IoT device receives a beacon message from the smartphone, it acknowledges the beacon message by sending the smartphone a response message with its device identifier and device information (e.g., device category, vendor name, and device model). The smartphone records this response message along with its position coordinate and the message's received signal strength as trace data for the localization. After a couple of hours, the recorded trace data contain the sample points of the smartphone near an IoT device.

SALA uses the correlation that the closer distance between two devices leads the stronger received signal strength of the exchanged message frame. The sample points with the received signal strength are processed to identify a grid position for the IoT device with the highest possibility. The contributions of this paper are as follows.

- An indoor localization architecture based on smartphone. This paper proposes an architecture using a smartphone-assisted localization algorithm called SALA for IoT devices in indoor environments. The architecture consists of a smartphone as a mobile anchor node with self-tracking capability and a server to perform offline localization algorithms along with the layout information of the indoor environment.
- A power map algorithm. A smartphone collects sample points for IoT devices through short-range communication. The power map algorithm constructs a power map whose entries have the fields of a device?'s identifier, the reception power for a message transmitted by the device, and the estimates (i.e., mean and standard deviation) of the distance between the smartphone and the device.
- A grid-weight map algorithm. With the power map, the grid-weight map algorithm associates a device's estimate position with a grid point. All grid points become to get their weight for all sample points by using the accumulative weight function through the Gaussian distribution model for the distance between each sample point and the device's estimated location.

The remaining structure of the paper is as follows. Section 2 explains related work. Section 3 specifies our problem formulation for indoor IoT device localization. Section 4 describes the design of our localization algorithm called SALA. Section 5 evaluates the performance of our SALA and two baselines. Section 6 concludes this paper along with future work.

2 Related work

Many localization schemes have previously been proposed for mobile ad-hoc networks and wireless sensor networks for outdoor localization. They can be categorized into the following three classes: (1) Range-based localization schemes [14, 15], (2) Range-free localization schemes [16–20], and (3) Event-driven localization schemes [21]. These outdoor localization schemes cannot be used for indoor localization where GPS signals are unavailable or weak. However, many indoor localization schemes use the basic ideas from the outdoor localization schemes. Also, there are three major approaches for indoor localization: (1) WiFi fingerprint [22–25], (2) Range-based localization [8, 9], and (3) Dead reckoning [3, 10, 13]. Thus, in this section, we review the literature of outdoor localization schemes, and then that of indoor localization schemes.

Range-based localization schemes use ranging devices (e.g., ultra-sonic sensor and laser) to measure signal arrival time difference and angle difference for distance measurement between adjacent devices. For the signal arrival time difference, Time Difference Of Arrival (TDOA) [14] and Angle Of Arrival (AOA) [15] are proposed.

Range-free localization schemes use message exchanges to estimate distance between adjacent devices. DV Hop and its deviations are well-known range-free localization algorithms [16–19]. The DV Hop algorithm works well in case of isotropic networks where anchor nodes with GPS receiver are uniformly placed in the entire area of interest. A range-free localization scheme beyond network connectivity is proposed to improve localization by using signature representing relative distances from each node to its neighboring nodes [20]. However, in this paper, we use only smartphones for localization as a mobile anchor node or beacon node indoors where GPS signal is not available.

Event-driven localization schemes use artificial event generation and event detection sequence for the localization. Zhong et. al propose a node-sequence-based localization scheme that uses the sequences of anchor nodes sorted by the received signal strength [21]. They first divide a map into small areas and calculate all node sequences for event detection. Each time, the target device chooses an area which has the best similarity between the area and its node sequences.

Many indoor localization schemes have recently been proposed with WiFi, beacons (or anchors), and smartphones [8, 9, 13]. However, these indoor localization schemes require either the expensive training for system deployment (e.g., WiFi signal fingerprint) or additional infrastructure (e.g., anchors and ranging devices). On the other hand, our SALA does not require such additional infrastructure and expensive

training. SALA can support the indoor localization of IoT devices, such as at home, in an autonomous way. This is possible because SALA uses short-range communication between the smartphone and IoT devices in order to get the proximity of the IoT devices indoors.

There are many fingerprint-based indoor localization schemes, which use fingerprint map to position a device. They use the deployed APs and collect WiFi Received Signal Strength Indicator (RSSI) data to make a fingerprint map. But at least five or six APs and prior sensing data are required to get a reliable map [22, 23]. However, based on the fact that FM uses channels different from ISM band used by WiFi, FM-based localization generates a better fingerprint map [24, 25].

He et al. [5] proposed a WiFi-based indoor localization scheme for tracking smartphones in indoor environments. This scheme uses a Gaussian process regression in order to train the collected dataset of WiFi RSS, and also uses a particle filter in order to estimate the smartphone's location. Like other WiFi-based fingerprints, this scheme constructs discrete location fingerprints during offline training phase and calculates the estimate of each location by the Bayesian decision rule along with the observed signal strength. Since a smartphone can move along any path, the collected fingerprints are interpolated by a Gaussian process and the location of the smartphone is determined by a particle filter. Like other WiFi-based fingerprints, since this scheme requires multiple reference points (i.e., APs) with training phase as pre-configuration, it cannot be used at apartments with one or two APs.

Jiang et al. [7] proposed a distributed RSS-based localization scheme for indoor IoT devices. This scheme uses a dynamic circle expanding mechanism that can accurately establish the geometric relationship between an unknown node and reference nodes for localization. This dynamic circle expanding mechanism can flexibly expand the coverage of localization circles that are based on the confidence level of three reference nodes until the circles of the reference nodes intersect each other. This mechanism is based on the fact that the RSSI becomes more unreliable as the distance from the signal source increases. The distance is modeled as multiple stages according to the range of RSSI. An unknown node discovers neighboring reference nodes by broadcasting discover packets and computes its coordinate according to the distances translated from the RSSIs with the reply packets from at least three reference nodes by using the dynamic circle expanding mechanism. Since the RSSI-distance relationship depends on the layout of indoor environments, a model for the RSSI-distance relationship cannot be generally for all indoor environments. Also, because this scheme requires references points, it cannot be used for apartments having one or two WiFi APs.

Otsason et al. [7] proposed a GSM-based indoor localization scheme where GSM stands for Global System for Mobile Communications for 2G cellular networks. This scheme uses wide signal-strength fingerprints that are constructed by 6-strongest signal cells and more available cells. They claimed that since the signals from the cells were more stable than those from WiFi APs, their proposed scheme was more stable and robust than WiFi-fingerprint-based schemes. However, this scheme cannot be used in rural areas with a less number of cellular base stations and requires overhead for training phase like other WiFi-fingerprint-based schemes.

Chintalapudi et al. [4] proposed an indoor localization scheme without pre-deployment effort, such as the construction of a fingerprint map through RSS measurements from WiFi APs. The scheme uses the EZ localization algorithm that leverages the constraints of physical wireless propagation and utilizes a genetic algorithm to solve them for localization. Even though this scheme does not require any explicit pre-deployment calibration, it yields higher localization errors from 2 to 7 m. However, in environments with residential areas with a small number of APs, this scheme cannot provide a reasonable indoor localization.

By dead reckoning, smartphones can have self-tracking capability in indoor environments where GPS signal is unreachable. In [3, 10], it is shown that such self-tracking capability can be implemented by a smartphone's motion sensors, such as accelerometer, gyroscope sensor, and magnetometer. In this paper, we use this self-tracking capability to localize IoT devices including appliances in smart home environments.

Hsu et al. [10] proposed a smartphone-based indoor localization scheme. This scheme is based on dead reckoning using the smartphone's motion sensors, such as an accelerometer and a gyroscope. The accelerometer's sensing data are used for step count, that is, measuring how many steps the smartphone user took. The gyroscope's sensing data are used for direction change, that is, measuring how much the smartphone user turned to the right or left. Since the motion error is accumulated over time, this scheme uses calibration marks to correct this accumulated motion error. This scheme's dead reckoning does not use the layout information, that is, apartment structure. On the other hand, our SALA's dead reckoning uses particle filtering based on the layout information, it can support more accurate self-position-tracking than Hsu's proposed one.

Jun et al. [13] proposed an enhanced indoor localization scheme called Social-Loc with social sensing. To cooperatively calibrate the estimation errors in localization, Social-Loc exploits encounter and non-encounter events of moving smartphones that performs self-tracking. Social-Loc aims at the localization of a moving smartphone user

with multiple cooperative smartphones rather than the localization of static IoT devices with a single smartphone.

Liu et al. [11] proposed a Hybrid smartphone Indoor Positioning Engine (called HIPE) for mobile location-based services. HIPE is a hybrid approach fusing a smartphone's motion sensors with WiFi fingerprints. The smartphone's motion sensors measure the user's motion dynamics information, such as step count (by an accelerometer) and heading direction (by a digital compass). To perform the fusion of motion information for position estimates, a hidden Markov model is used along with a grid-based filter algorithm and the Viterbi algorithm. HIPE does not use indoor layout information, so it can be improved more like SALA's dead reckoning. HIPE is for the tracking of a smartphone's position, not for the localization of IoT devices near the smartphone, which is SALA's objective.

Range-based indoor localization schemes have been proposed to improve WiFi fingerprint that tends to have the weak signatures from the unstable RSS over time [8, 9]. Liu et al. [8] proposed an accurate WiFi based localization with smartphone-based peer ranging in indoor environments. They showed the limitation of pure WiFi-based methods, which has the ambiguity of similar signatures among distinct locations. They improved the accuracy of WiFi-based localization through acoustic ranging estimates among peer smartphones and the mapping of their locations against a WiFi signature map that is subject to ranging constraints. However, since their scheme requires multiple APs for WiFi signatures and many smartphones for acoustic ranging, it is not feasible for the localization of IoT devices at home.

As another range-based localization scheme, Guoguo was proposed as an indoor localization ecosystem, using acoustic ranging among anchors and smartphones [9]. Their scheme uses low transmission power for the anchor network longevity and also acoustic, unnoticeable beacons for sound pollution avoidance. Though their scheme can provide centimeter-level localization accuracy for smartphone positioning, it requires anchor nodes for acoustic ranging with smartphones and the manual localization of these anchor nodes. This setting is overkill for IoT device localization at home. On the other hand, our SALA does not require such anchor nodes and manual localization, so it is a more feasible approach at home environments.

Therefore, through the literature review, the legacy indoor localization approaches are focused on the localization of mobile nodes (e.g., smartphones) rather than their neighboring nodes indoors. In this paper, we propose SALA for the cost-effective localization of indoor IoT devices by using smartphones as mobile anchors along with the short-range communication among the smartphones and IoT devices.

3 Problem formulation

In this section, we specify assumptions for the localization at a target indoor place (e.g., home) and describe our localization problem along with the objective in this paper. For a given home layout with IoT devices, the objective is to localize the positions of the IoT devices through the short-range communication between a smartphone and each IoT device, indicating at which part each IoT device is located, such as open space, wall or corner, in a specific room in the target indoor place.

3.1 Assumptions

We have the following list of assumptions for SALA:

- The layout of a target place (e.g., home) is available to a smartphone in the format of a digital map. Fig. 2 shows the layout of an apartment with IoT devices, such as smart TV, refrigerator, washer machine, CD player, air conditioners, and robot cleaner. In this paper, we target at the localization in an apartment with a single floor. The localization of a house with multiple floors is feasible by extending our SALA in three dimensions. The localization in three dimensions is left as future work.
- A smartphone has self-tracking capability to track its position at home with motion sensors, such as accelerometer, gyroscope, and magnetometer from the entrance of the target home, as shown in Fig. 2 [3, 12, 13]. This self-tracking is enabled with step counter based on accelerometer and direction measure based on magnetometer and gyroscope.
- A smartphone and IoT devices can communicate with each other within short distance through the transmission power control of wireless communications, such as WiFi, RFID, ZigBee, or NFC. In this paper, it is

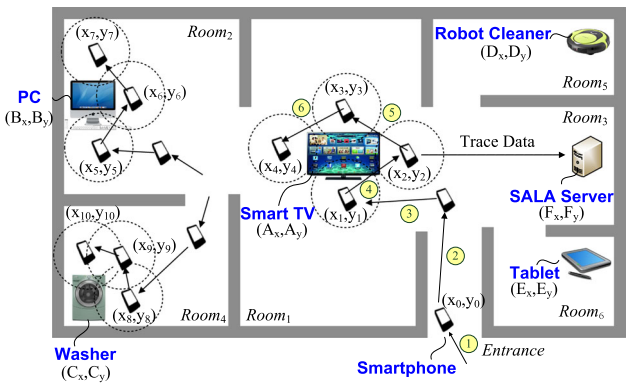


Fig. 2 SALA Localization for positioning IoT Devices. The positions of the IoT devices can be estimated by using a smartphone as a mobile anchor and short-range communication with IoT devices

assumed that they can communicate with each other in WiFi ad-hoc mode. When the smartphone broadcasts a beacon message for localization, each IoT device can respond to the message with a reply message, containing the IoT device's information, such as identifier, vendor and model. This interaction between the smartphone and IoT devices is active probing. Along with this active probing, SALA can support the localization with passive probing where the RSSI of the messages from IoT devices to an AP at home is measured. The performance of both active probing and passive probing will be evaluated in Sect. 5. The smartphone stores the captured device information into its local repository along with the reception timestamp and RSSI of the reply message and the coordinate of the smartphone.

- There is a dedicated server (called SALA Server) at home that performs localization with trace data collected by the smartphone. The trace data consist of the tuples of timestamp, smartphone's position, device information, and RSSI. Fig. 2 shows SALA Server, collecting trace data from a smartphone.
- A smartphone can recognize its entrance into a target apartment to start tracing its trajectory (i.e., movement path) indoors. This assumption can be realized by a fusion sensing with the Service Set Identifier (SSID) and signal strength of a WiFi AP at home [26] and the sound detection of the entrance door opening [27]. Also, a beacon having home identification can be used to give the entrance information to the smartphone [28].

3.2 Concept of localization in SALA

In this section, we describe the concept of localization in SALA with Fig. 2. First of all, we categorize the positions of IoT devices in Fig. 2 into the following three classes: (1) Open space, (2) Wall, and (3) Corner. As an open-space IoT device, a smart TV is located in the middle of $Room_1$. As wall IoT devices, a desktop PC and a home server (called SALA Server) are located near the wall of $Room_2$ and $Room_3$, respectively. As corner IoT devices, a washer, a robot cleaner, and a tablet are located at the corner of $Room_4$, $Room_5$, and $Room_6$, respectively. SALA can localize the positions of these IoT devices wherever they are located, such as open space, wall, and corner with trace data of a smartphone, as shown in Fig. 2.

Now we explain the procedure of SALA localization at an apartment in Fig. 2. When a smartphone detects the position of the entrance at home, it starts to trace its trajectory. It stores the initial position (marked as 1) as (x_0, y_0) along with timestamp t_0 into its local repository in the relative coordination system at home. The smartphone moves along the trajectories denoted as 2 and 3 and reaches the position of (x_1, y_1). At this position, it broadcasts a beacon frame with timestamp to a smart TV and gets the response from the smart TV. It stores position (x_1, y_1) with timestamp t_1 into its local repository. In the same way, it stores positions (x_2, y_2), (x_3, y_3), and (x_4, y_4) along with timestamps t_2, t_3, and t_4, respectively, while it moves along the trajectories denoted by 4, 5, and 6. For the same IoT device, such as smart TV, the smartphone has four coordinates, as shown in Fig. 2. With these four coordinates, we can calculate a centroid as an estimate of the smart TV's location. The challenge in the localization is that these four coordinates may have different distances from the ground-truth location of the smart TV because of environmental effects, such as WiFi radio irregularity and multi-path fading. That is, the communication range limited by transmission power control is not constant, such as 2 m. To deal with this challenge, we use a statistical approach with many sample coordinates obtained from short-range communication between the smartphone and IoT devices. In the next section, we will explain the design of our SALA system along with its core algorithms.

4 Design of SALA System

This section explains the design of our Smartphone-Assisted Localization Algorithm (SALA) and the justification of our design.

4.1 SALA system architecture and localization procedure

This subsection explains the system architecture and localization procedure of SALA. Fig. 3 shows our SALA system architecture. As shown in the figure, SALA system consists of two nodes, such as smartphones and SALA Server. The localization of IoT devices is performed by the following seven steps:

1. *IoT device detection* Whenever a smartphone s_k detects an IoT device d_i by short-range communication at time t_j^i at its Cartesian coordinate p_j^i with the power of the reply message w_j^i from the IoT device d_i, s_k stores this detection into its local repository. When s_k collects a sufficient number of coordinates for d_i, s_k reports the set C_i of tuples (t_j^i, p_j^i, w_j^i) for d_i to SALA Server.
2. *Collection of IoT device detection trace* SALA Server collects IoT device d_i's trace data C_i that are a set of tuples (t_j^i, p_j^i, w_j^i) where t_j^i is a timestamp, p_j^i is a detection position, and w_j^i is the RSSI of the reply message from d_i.

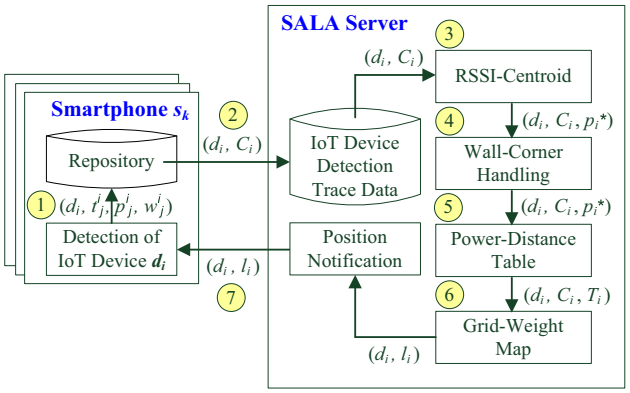


Fig. 3 SALA System Architecture. Localization procedure consists of (*1*) IoT device detection, (*2*) collection of trace data, (*3*) RSSI-centroid estimation, (*4*) wall-corner handling algorithm, (*5*) power-distance table construction, (*6*) grid-weight map construction, and (7) position notification

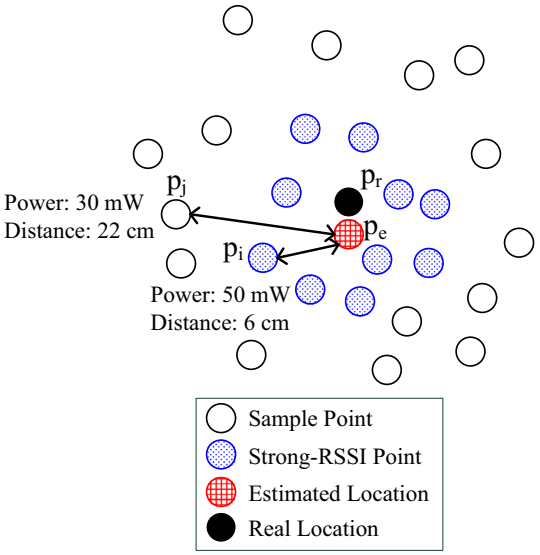


Fig. 4 Distance and power measurement. The figure shows how the sample points for RSSI-centroid estimation are selected

3. *RSSI-centroid estimation* SALA Server performs RSSI-Centroid estimation to compute an initial estimate p_i^* of the IoT device d_i's position from the trace data C_i.
4. *Wall-corner handling algorithm* When an IoT device is located at either wall or corner, sample points may not surround an IoT device geographically. In this case, the grid-weight map cannot make an accurate estimation of the IoT device d_i. The skewed distribution of sample points will be handled, taking advantage of the layout information for the localization of such an IoT device d_i.
5. *Power-distance table construction* SALA Server constructs a power-distance table T_i where an entry for each sample point $p_j \in C_i$ has a pair of the distance between p_j and p_i^* and the RSSI for d_i at p_j.
6. *Grid-weight map construction* From the power-distance table T_i, a grid-weight map is constructed. The map of a target indoor area is divided into grid cells whose centers are used as the location estimates of an IoT device. With T_i, each grid cell will be assigned a weight value per IoT device d_i.
7. *Position notification* Once the IoT device d_i is localized by SALA Server, the location information l_i is delivered to d_i.

With exemplary figures, we explain the procedure of SALA localization in a target localization place, such as an apartment in Fig. 11. As shown in Fig. 4, during (1) IoT device detection and (2) the collection of IoT device detection trace, a smartphone s_k collects sample points for an IoT device d_i with its real location p_r, and delivers them to SALA server for the localization of d_i. After the collection of sample points for d_i, through (3) RSSI-centroid estimation, SALA server computes an initial estimated location p_i^* (also called p_e) of d_i, as shown in Fig. 9a where the estimated location p_e is a little far away from the real location p_r. In the case where the estimated location p_e is close to either wall or corner, as shown in Fig. 5, (4) wall-corner handling algorithm makes artificial sample points for a better estimation of the real location p_r of d_i. Through (5) power-distance table construction, as shown in Table 1, a power-distance table is constructed to contain the ordered tuples of sample points for a robust distance estimate for the real location p_r and the reception radio power from p_r. Through (6) grid-weight map construction, as shown in Fig. 6, a grid-weight map is constructed to put an

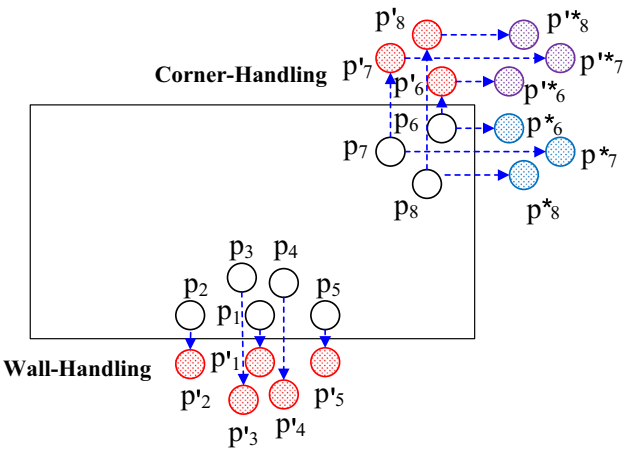


Fig. 5 Wall-Corner Handling for special devices on the wall or at the corner. This procedure checks whether an IoT device is located near a wall or corner. If so, it reflects the sample points on the opposite side(s) of the wall or corner as virtual sample points

Table 1 Power-distance table

Metrics/Point	p_1	...	p_{k-1}	p_k	p_{k+1}	...	p_n
Power	w_1	...	w_{k-1}	w_k	w_{k+1}	...	w_n
Distance	l_1	...	l_{k-1}	l_k	l_{k+1}	...	l_n
Avg. Distance	μ_1	...	μ_{k-1}	μ_k	μ_{k+1}	...	μ_n
Std. Deviation	σ_1	...	σ_{k-1}	σ_k	σ_{k+1}	...	σ_n

Power is the RSSI value. Distance is the Euclidean distance between a sample point and the estimated location of an IoT device. Avg. Distance is the average of the distances of multiple neighboring sample points for the estimation location. Std. Deviation is the standard deviation of the distances of those sample points for Avg. Distance

accumulative weight to each grid in the target localization place. A grid point l_i with the highest weight is regarded as the final location estimate of the real location p_r, as shown in Fig. 9d. Finally, in (7) position notification, the final location estimate l_i is delivered to the IoT device d_i by SALA server.

So far we have explained the system architecture and localization procedure of SALA. In the next subsection, main localization algorithms in Steps (3)$\sim$(6) will be explained.

4.2 Localization algorithms

In this section, component algorithms for SALA localization are explained, such as (1) RSSI-Centroid Estimation, (2) Wall-Corner Handling, (3) Power-Distance Table Construction, and (4) Grid-Weight Map Construction.

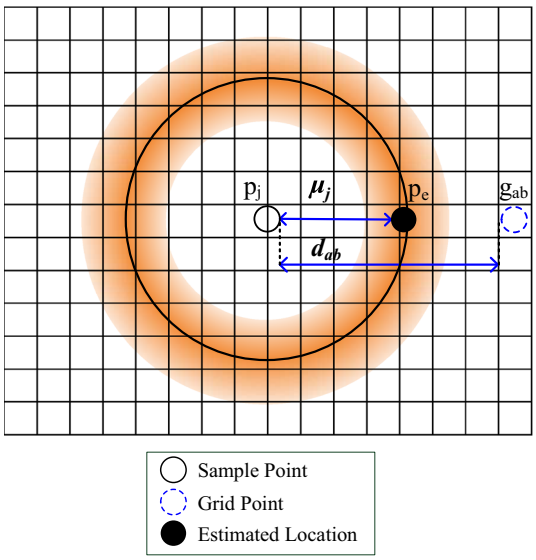


Fig. 6 Grid-Weight Map constructed by Power-distance table. This map is used in the tuning for the better estimation of the IoT device's position

4.2.1 RSSI-centroid estimation

Assume that sample points are distributed around the real location of an IoT device d_i, as shown in Fig. 4. Our RSSI-Centroid estimation makes an initial position estimate (called estimated location) of an IoT device d_i from the sample points in Fig. 4. Let C_i be a set of sample tuples (t_j, x_j, y_j, w_j) for IoT device d_i where t_j is a timestamp, x_j is x-coordinate, y_j is y-coordinate, and w_j is RSSI value. Let the set size $|C_i|$ be n. We sort the sample tuples into a list C_i' according to the descending order of RSSI values w_j such that $C_i' = \{p_1, p_2, \ldots, p_{k-1}, p_k, p_{k+1}, \ldots, p_n\}$. We select the first m tuples in the sorted list C_i' and make a new list C_i^* where m is usually set to 10. We then compute a centroid of these m sample points in C_i^* as follows:

$$x_i^* = \frac{1}{m} \sum_{x_k \in C_i^*} x_k, \qquad y_i^* = \frac{1}{m} \sum_{y_k \in C_i^*} y_k. \tag{1}$$

From (1), let $p_i^* = (x_i^*, y_i^*)$ be an initial position estimate of the IoT device d_i. When the sample points surround the IoT device d_i, this RSSI-Centroid p_i^* gives us an accurate position estimate of d_i. This p_i^* is used as an initial position estimate to construct a power-distance table for a more accurate position estimate in the next step.

4.2.2 Wall-corner handling

Differently from an open-space IoT device, which allows sample points to surround the IoT device, we treat the case where an IoT device is located near either wall or corner. Such an IoT device is called *wall-corner IoT device*. In this case, the sample points are not surrounding the IoT device, as shown in Fig. 5, leading to an inaccurate initial position estimate of the IoT device that is far away from the real location of the IoT device. Thus, in this subsection, we handle these wall-corner IoT devices such that the sample points surround them by adding virtual sample points that are artificial sample points. As a result, when the sample points surround a wall-corner IoT device, the initial position estimate based on centroid is usually close to the real position of the IoT device.

First of all, we check whether an IoT device is located near wall or corner. The initial position estimate p_i^* is found by RSSI-Centroid estimation in Sect. 4.2.1. In a wall case, if p_i^* is located near wall (e.g., 2 meters from the wall), we select the first x sample points (e.g., p_1, p_2, p_3, p_4, and p_5 in

Fig. 5) in the RSSI-based-sorted list C_i'. We flip them to the wall, getting virtual sample points (e.g., p_1', p_2', p_3', p_4', and p_5' in Fig. 5). In a corner case, if p_i^* is located near corner (e.g., 2 meters from the corner), we flip the sample points (e.g., p_6, p_7, and p_8 in Fig. 5) to both two walls adjacent to the corner in one step at a time. These virtual sample points include single flipped ones and double flipped ones. In Fig. 5, the single flipped ones are p_6', p_7', and p_8' (or p_6^*, p_7^*, and p_8^*) in Fig. 5). In the figure, the double flipped ones are $p_6'^*$, $p_7'^*$, and $p_8'^*$. They are merged to C_i'. This augmentation of virtual sample points can reduce the impact of the biased sample points on the skewedness of the initial position estimate in Power-Distance Table, discussed in Sect. 4.2.3.

4.2.3 Power-distance table construction

In this step, a power-distance table is constructed as a data structure to construct a grid-weight map in the next step. This step uses the RSSI-based-sorted list C_i'. The initial position estimate p_i^* is computed by the centroid of the points in C_i' by Equation (1) where $p_i^* = (x_i^*, y_i^*)$. Let p_j be a sample point in C_i'. We make an Euclidean distance $dist(p_i^*, p_j)$ between p_i^* and $p_j \in C_i'$. In Fig. 4, the centroid p_i^* is denoted as p_c.

Table 1 shows a power-distance table made from the sorted sample set C_i^* and its centroid p_c. In the table, a point p_k has four values, such as power (w_k), distance (l_k), average distance (μ_k), and distance deviation (σ_k). The power w_k is the RSSI value for the IoT device d_i at the point p_k. The distance l_k is the Euclidean distance between the centroid p_c and the point p_k. The average distance μ_k is the average of the distances of multiple points near the point p_k, e.g., 10 samples that consist of one sample p_k, four samples left to p_k, and five samples right to p_k. This average distance is regarded as the estimated distance between p_k and the real position (denoted as p_r in Fig. 4) of the device d_i. Most of points p_k can take their left four neighboring points and right five neighboring points for the average distance. However, the leftmost point p_1 takes its right five neighboring points and the rightmost point p_n takes its left four neighboring points for the average distance.

The rationale of the average distance is that the distance l_k between the sample point p_k and the centroid p_c has noise due radio irregularity [29] in the measurement of RSSI values. Thus, m strongest RSSI-value points are selected for a stable centroid p_c. By simulation, we know that the average distance μ_k is a more reliable distance metric than distance l_k. In the same way as the average distance μ_k, we can compute the standard deviation σ_k for m sample points including p_k and the estimated location p_e with the distances of multiple points near the point p_k for the average distance μ_k.

Let p_e be an estimated location for device d_i from the power-distance table. In Fig. 6, p_e is μ_j away from a sample point p_j. Note that the position of p_e will be estimated from the grid-weight map in the next subsection. We have so far constructed a power-distance table for sample points in C_i' and the centroid p_c. In the next subsection, we construct a grid-weight map with the power-distance table that is used to estimate the real location of the IoT device d_i.

4.2.4 Grid-weight map construction

A grid-weight map is constructed with the power-distance table for the estimation of the real location p_r of the IoT device d_i. Fig. 6 shows a grid-weight map that is constructed by a sample point p_j and its average distance μ_j to the estimated location p_e in Table 1. As shown in the figure, the estimated location of the IoT device d_i is regarded as being on the perimeter of the circle whose center is p_j and whose radius is μ_j. Next, we give each grid cell a weight corresponding to the probability that the grid cell has the IoT device d_i. Such a probability is determined by the probability distribution that is a normal distribution $N(\mu_j, \sigma_j)$ where μ_j is the average distance and σ_j is the standard deviation for the sample point p_j in Table 1. Let N_{row} be the number of rows and N_{col} be the number of columns in the grid-weight map M. Let g_{ab} be a grid cell where a is the row index in $[1, N_{row}]$ and b is the column index in $[1, N_{col}]$, as shown in the grid-weight map M in Fig. 6. Let d_{ab} be the shortest distance between the grid cell g_{ab} and p_j, as shown in Fig. 6. Let f_{ab} be the probability of the normalized value $\frac{d_{ab}-\mu_j}{\sigma_j}$ in the standard normal distribution $N(0, 1)$ (i.e., Gaussian distribution), as shown in Fig. 7. Note that Fig. 8 shows a three-dimensional probability distribution for a grid-weight map for a given sample

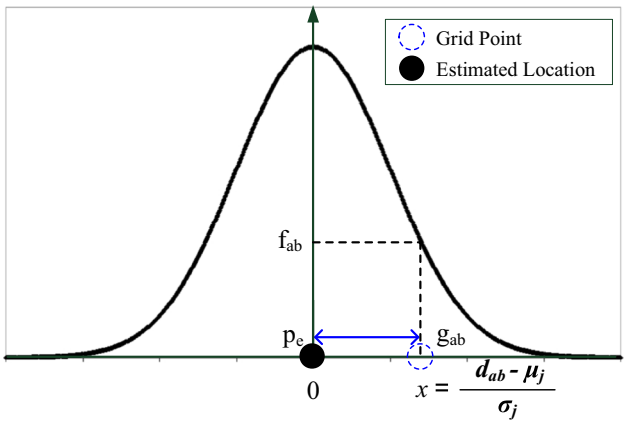


Fig. 7 Probability distribution for grid-weight map. This is a Gaussian distribution used for calculating the cumulative grid-weight for each grid cell

point p_j. This figure is the combination of the grid-weight map in Fig. 6 and the probability distribution in Fig. 7. Let c_{ab} be the cumulative grid-weight for the grid cell g_{ab}, which is initially set to 0.

Algorithm 1 Grid-Weight Map Computation Algorithm

1: **function** $Grid\_Weight\_Map(C,T)$ ▷ Input: C is sample point set and T is power-distance table
2: $M = (m_{ab}) \leftarrow 0$ for $a = 1, ..., N_{row}$ and $b = 1, ..., N_{col}$ ▷ initialize the grid-weight matrix $M = (m_{ab})$ with 0
3: **for all** $p_j \in C$ **do**
4: $[l_j, \mu_j, \sigma_j] \leftarrow Get\_Table\_Entry(T, p_j)$ ▷ get the table entry values l_j, μ_j, σ_j for sample point p_j
5: **for** $a \leftarrow 1, N_{row}$ **do**
6: **for** $b \leftarrow 1, N_{col}$ **do**
7: $d_{ab} \leftarrow Get\_Distance(p_j, g_{ab}, l_j)$ ▷ compute the Euclidean distance between a grid cell g_{ab} and the perimeter of the circle whose center is p_j and whose radius is μ_j
8: $f_{ab} \leftarrow Get\_Probability(d_{ab}, \mu_j, \sigma_j)$ ▷ compute the probability of g_{ab} with the distance d_{ab} between p_j and g_{ab} with the normal distribution $N(\mu_j, \sigma_j)$
9: $m_{ab} \leftarrow m_{ab} + f_{ab}$
10: **end for**
11: **end for**
12: **end for**
return M
13: **end function**

To construct the grid-weight map M, for each sample point $p_j \in C_i$, the weight f_{ab} is computed for each grid cell g_{ab} and is added to c_{ab}. Algorithm 1 specifies the whole procedure to compute the grid-weight map M for the sample point set C. In line 2, the grid-weight matrix M is initialized with zeroes. In lines 3–12, with sample points in C, the cells in the grid-weight matrix M become to have weight for the location estimation of each IoT device d_i. Note that a grid cell with the greatest weight is estimated as a real location candidate with the highest probability. In line 4, *Get_Table_Entry* computes the table entry values of distance l_j, average distance μ_j, and standard deviation σ_j for each sample point p_j. In lines 5–11, for each sample point $p_j \in C$, the cells in M are populated with accumulated weight for the location estimation of an IoT device d_i. In line 7, *Get_Distance* computes the Euclidean distance between a grid cell g_{ab} and the sample point p_j whose radius is μ_j in Fig. 6. In line 8, *Get_Probability* computes the probability (denoted as f_{ab}) of g_{ab} with the distance d_{ab} between p_j and g_{ab} with the normal distribution $N(\mu_j, \sigma_j)$. In line 9, the cell m_{ab} becomes to have the accumulated value of $m_{ab} + f_{ab}$.

The time complexity of Algorithm 1 is $O(|C| * N_{row} * N_{col})$. Algorithm 1 has triple loops where the first loop has $|C|$ iterations, the second loop has N_{row} iterations, and the third loop has N_{col} iterations.

After running Algorithm 1, the center of a grid cell g^*_{ab} with the maximum weight in M is estimated for the real location p_r of the IoT device d_i, as shown in Fig. 4. If there are more than one grid cell with the maximum weight, the centroid of those grid cells is estimated for the real location p_r of the IoT device d_i. Fig. 9 shows the searching sequence for the real location of an IoT device d_i through Algorithm 1 along with a grid-weight map. At Step 1, there is a non-negligible distance between an estimated location

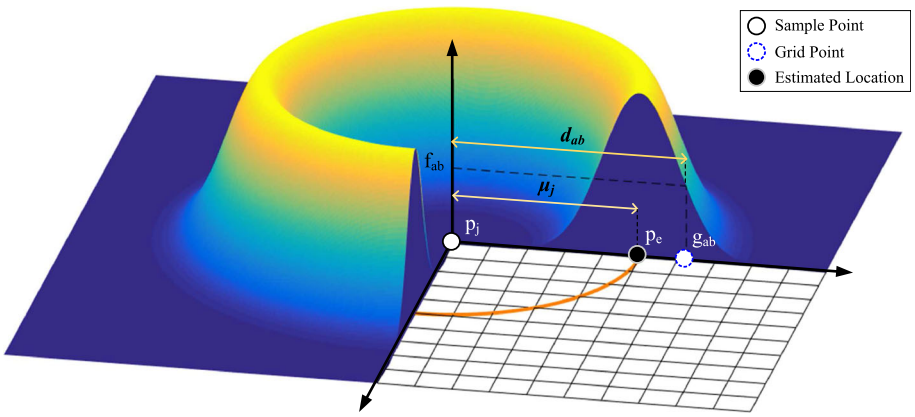


Fig. 8 Three-dimensional probability distribution for grid-weight map. The grid cells near the estimated location have higher values than others by the property of a Gaussian distribution

and the real location. During the iterations of Algorithm 1, it can be observed that the estimated location is getting closer to the real location.

So far we have explained our SALA algorithm through RSSI-Centroid estimation, power-distance table, and grid-weight map. In the next section, we will evaluate our SALA in realistic home settings along with two baseline localization schemes.

5 Performance evaluation

This section evaluates the performance of SALA in terms of localization error and localization accuracy of IoT devices. The evaluation setting is as follows:

- *Performance metrics* We use two metrics: (1) *localization error* is defined as the Euclidean distance between the real location and estimated location of an IoT device. (2) *localization accuracy* is defined as the accuracy percentage value for room-level localization estimating which room an IoT device is located at, such as kitchen, living room, study room, bedroom, and bathroom.
- *Parameters* We investigate the impact of the following parameters on the performance: (1) *Smartphone's moving time* and (2) *Smartphone's positioning error*.
- *Baselines* We make two baseline schemes since our SALA is the first work for the indoor localization of IoT devices using smartphone tracking that is based on a smartphone's motion sensors:

  (1) *RSSI* is a location estimation scheme for an IoT device that uses RSSI-Centroid algorithm in Sect. 4.2.1.

  (2) *Convex* is a location estimation scheme for an IoT device that uses the average of the circumcenters of three consecutive sample points on the convex hull for 10 strong RSSI sample points. This scheme also use sample points around an IoT device d_i like RSSI-Centroid algorithm, as shown in Fig. 10. First, Convex makes the set of the points that are included in the convex hull of the sample points into a *convex-hull point set* V_i, as shown in Fig. 10. Next, Convex gets three clockwisely consecutive points from V_i and computes the circumcenter of such points. Let U_i be the union of the circumcenters of all clockwisely consecutive points in V_i. Finally, we compute the *centroid* of the circumcenter points in U_i for the convex-hull point set V_i where the size of V_i is $n = |V_i|$ as follows:

$$
\begin{aligned}
x_i^{**} &= \frac{1}{n-2}\sum_{k=1}^{n-2} X\_Coordinate \\
&\quad (Circumcenter(p_k, p_{k+1}, p_{k+2})), \\
y_i^{**} &= \frac{1}{n-2}\sum_{k=1}^{n-2} Y\_Coordinate \\
&\quad (Circumcenter(p_k, p_{k+1}, p_{k+2})),
\end{aligned}
\tag{2}
$$

where $Circumcenter(p_k, p_{k+1}, p_{k+2})$ is the circumcenter of three points $p_k, p_{k+1}, p_{k+2} \in V_i$, and $X\_Coordinate(q_k)$ and $Y\_Coordinate(q_k)$ return the x-coordinate x_k and y-coordinate y_k of $q_k = (x_k, y_k)$, respectively. From (2), let $p_i^{**} = (x_i^{**}, y_i^{**})$ be a position estimate of the IoT device d_i by Convex method.

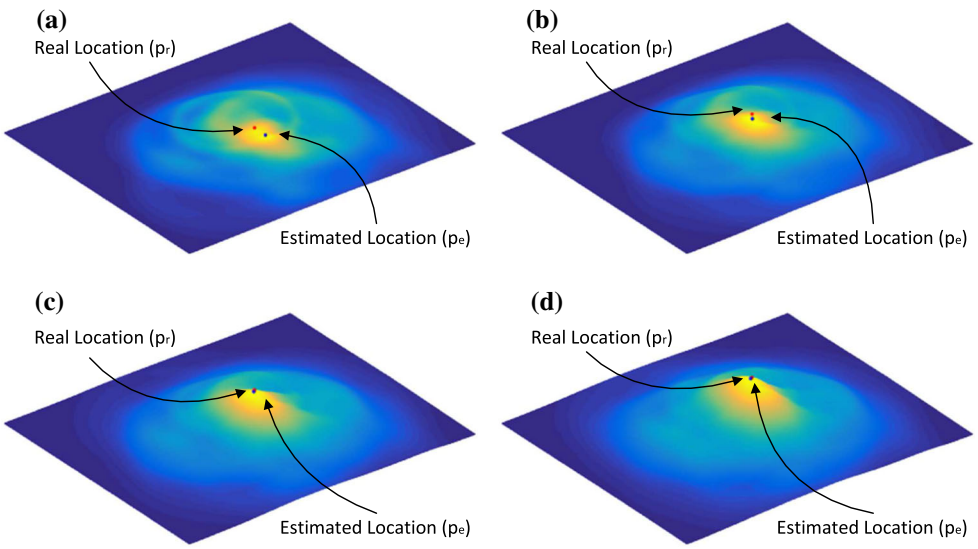


Fig. 9 Searching sequence for real location of IoT device. During the iterations of Algorithm 1, it can be observed that the estimated location is getting closer to the real location. **a** Step 1 (distance: 39.10 cm). **b** Step 2 (distance: 23.45 cm). **c** Step 3 (distance: 9.02 cm). **d** Step 4 (distance: 6.46 cm)

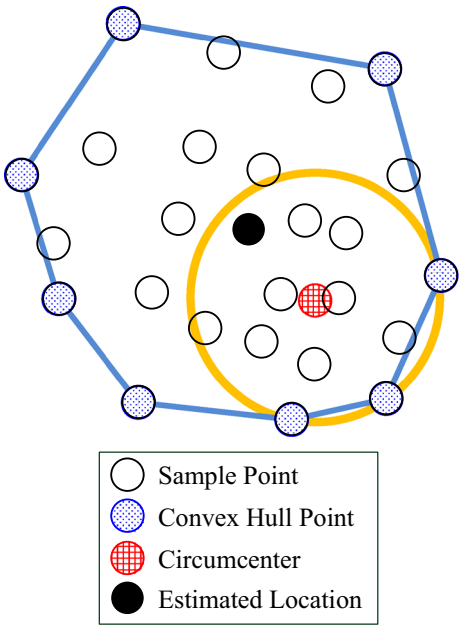


Fig. 10 Convex Method as a baseline localization scheme. Convex method takes three clockwisely-consecutive points and computes the circumcenter of such points as the estimated location of an IoT device

We have implemented our SALA and the two baselines with a popular network simulator called OMNeT++ [30]. For an indoor environment, we use a 109 m^2 apartment model, as shown in Fig. 11a. Ten IoT devices are placed in a 10.1 m $\times$ 9.3 m apartment except hall and veranda, as shown in Fig. 11b. Table 2 shows simulation configuration for the apartment in Fig. 11.

The IoT devices are located near to the open space, wall, or corner in the apartment as follows. Device 1 is located at the upper-left corner. Devices 2 through 7 are located close to the walls. Devices 8, 9, and 10 are located close to the corners of the entrances.

In the simulation, a smartphone moves with a random mobility, mimicking the mobility of a resident in an apartment. We assume that a smartphone can recognize entering the target apartment and start from the entrance as its initial position. Also, we assume that it can trace its trajectory by its motion sensors (e.g., accelerometer, gyroscope, and magnetometer). After it starts moving at the entrance of the apartment with its initial position, every two seconds, it chooses a next target point with a distance less than 2 m without any wall so that it can move to the target point with no collision.

Figure 12 shows how a smartphone has moved by a random mobility called random waypoint model [31]. Fig. 12a shows a mobility trajectory, that is, a moving path in the apartment. Fig. 12b shows a mobility distribution where each grid has a visiting frequency during the smartphone's travel. In this figure, a dark-colored grid has a relatively higher frequency than a light-colored grid. From a start point, the smartphone directly moves to a destination with a random velocity. Once it arrives at the destination, it selects another destination and moves towards the destination without any thinking time. It shows that the smartphone moved all around the house with enough sample points for the IoT devices.

For the collection of sample points in the localization period, we assume that the smartphone and IoT devices can communicate with each other via ad-hoc mode in WiFi. The smartphone transmits a probe request packet every 0.1 s. When an IoT device receives a probe request from the smartphone, the IoT device transmits its probe reply to the smartphone, containing the simulation time, the device ID, and the coordinate of the smartphone. We use Rician fading model as a radio propagation model, considering both line-of-sight signal and ground reflection.

We explain why Rician model is used as radio fading model in the simulation in comparison with Free-space model. In Free-space model, only path loss affects radio strength, so the shape of the convex hull for sample points is almost like a circle. However, in Rician model, the smartphone can communicate with an IoT device farther than the expected communication range R by the irregularity of wireless communication range [29]. However, SALA can reduce the impact of these outliers on the localization with many sample points around the communication range R from the IoT device.

5.1 Localization performance comparison

This subsection compares three localization schemes in terms of performance metrics, such as localization error, localization accuracy at apartment level, and localization accuracy at room level. Note that in this subsection, the localization error for each IoT device is the measurement for a single simulation run.

Figure 13 shows the localization error of ten IoT devices according to IoT device ID where about 200 sample points per IoT device are collected for two hours. In most cases, SALA has a smaller localization error than Convex and RSSI even though RSSI has better performance than SALA for IoT devices 2, 5, 6, and 8. In the term of the average of the localization errors for ten IoT devices, SALA outperforms both RSSI and Convex where SALA, Convex, and RSSI have the average localization error of 17.9, 53, and 23.4 cm, respectively.

Figure 14 shows the localization accuracy of ten IoT devices at the apartment level, that is, estimating at which room an IoT device is located. In figure, true value means that the corresponding scheme estimates the correct room where an IoT device is located. On the other hand, false value means that the corresponding scheme estimate a

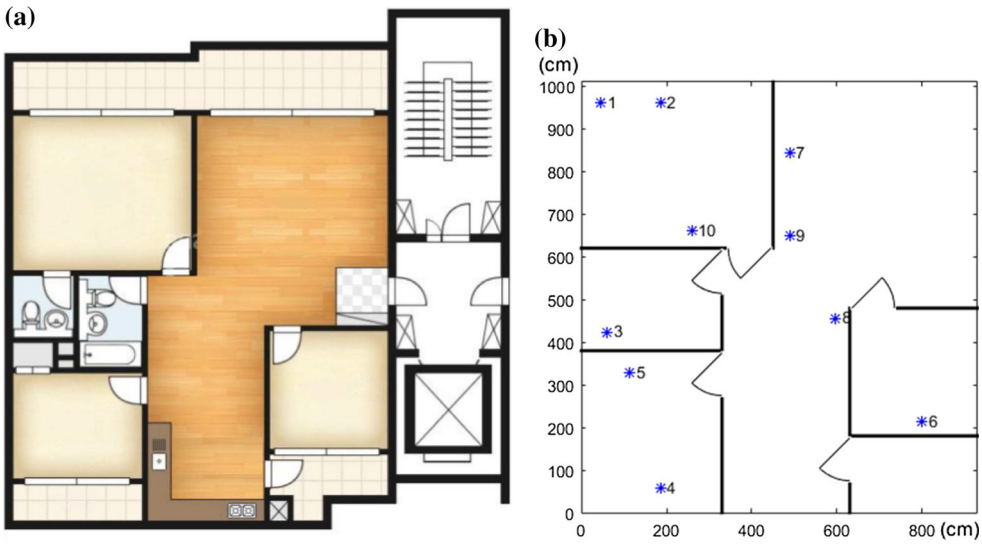


Fig. 11 Apartment (**a**) and IoT Device Deployment (**b**) for Simulation. A 109 m^2 apartment model is used with the layout of 10.1 m × 9.3 m. Ten IoT devices are placed in the apartment except the hall and veranda

Table 2 Simulation configuration for performance evaluation of three localization schemes

Parameter	Description
Apartment layout	An apartment that has three bedrooms, a kitchen, a living room, and two bathrooms with 10.1 m × 9.3 m (i.e., 39.7 feet × 38.4 feet).
Smartphone's moving time N	The time (in N minutes) when the smartphone has moved and collected data. The default of N is 120 min.
Smartphone position error ϵ	Smartphone's position error caused by motion sensors, such as accelerometer, gyroscope, and magnetometer. $\epsilon \sim N(0, \sigma)$ where the default of σ is 50 cm.

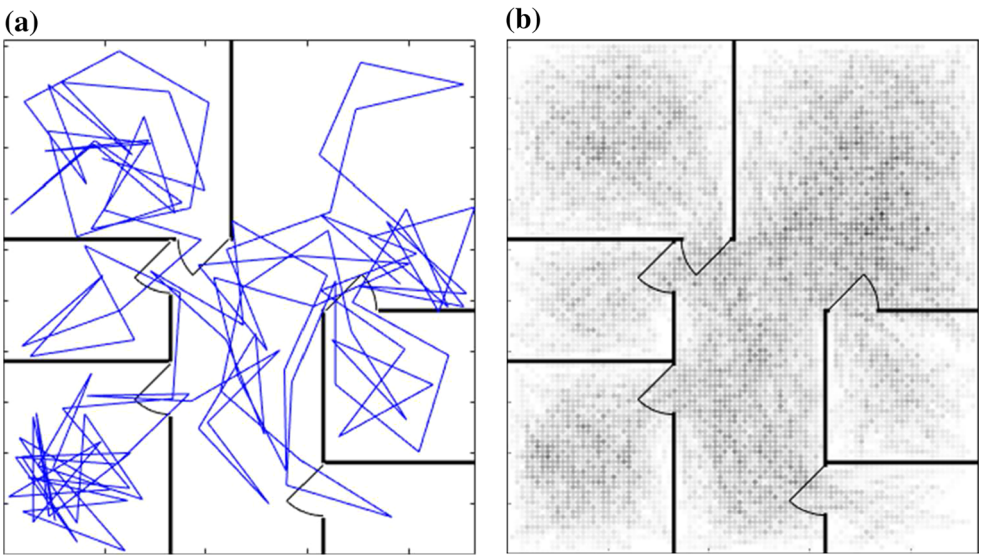


Fig. 12 Smartphone's indoor mobility. **a** shows the mobility trajectory, that is, the moving path of the smartphone in the apartment. **b** shows the mobility distribution where each grid has the visiting frequency of the smartphone during the smartphone's travel

wrong room for an IoT device. SALA can estimate the correct rooms of all the IoT devices, but RSSI and Convex estimate the incorrect room of an IoT device, respectively.

Figure 15 shows the localization accuracy of ten IoT devices at the room level, that is, estimating at which part in a room an IoT device is located, such as an open space,

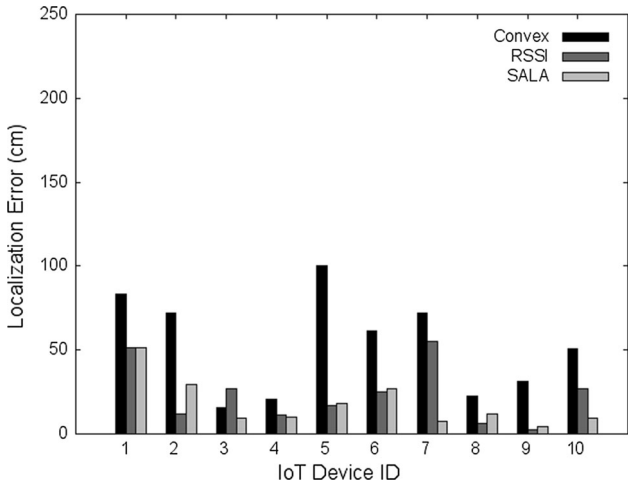


Fig. 13 Localization error of IoT devices. In most cases, SALA has a smaller localization error. The average localization error of SALA is 17.9 cm, while those of Convex and RSSI are 53 and 23.4 cm, respectively

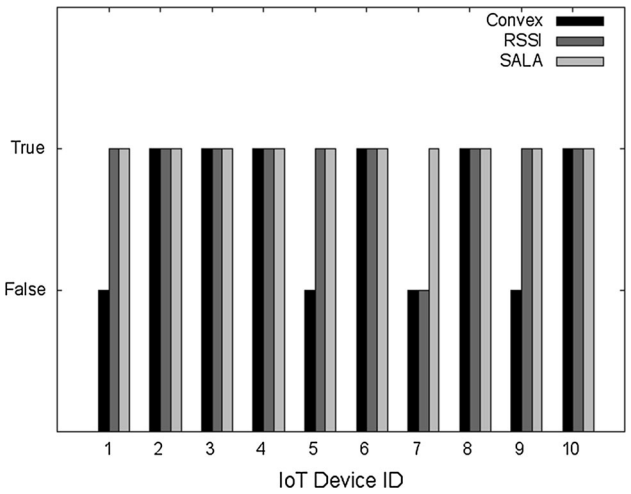


Fig. 15 Localization accuracy at room level. SALA can perfectly estimate the correct room-parts of all the IoT devices, while Convex and RSSI make some mistakes in the room-part estimation, respectively

wall, and corner. SALA can estimate the correct parts of all the IoT devices, but RSSI and Convex estimate the incorrect part of an IoT device, respectively. Thus, SALA has better performance than the other baseline schemes in most cases and perfect accuracy at both the room-level and apartment-level estimation.

5.2 Cumulative distribution function of localization error

This subsection shows the cumulative distribution function (CDF) of localization error for ten IoT devices. Fig. 16 shows the CDF of the localization errors in ten IoT devices in Fig. 13. SALA has a higher percentage for a low localization error (i.e., 50 cm) than both RSSI and Convex. That is, SALA allows 90 % IoT devices to have a low localization error of 50 cm. For a high localization error (i.e., at least 60 cm), RSSI has a little higher percentage than SALA. That is, SALA has some IoT devices with a little higher localization error than RSSI. Therefore, SALA lets most of IoT devices have lower localization errors than the other baseline schemes.

5.3 Impact of measurement time

To compare SALA with the other baseline schemes, we investigate the impact of measurement time on the

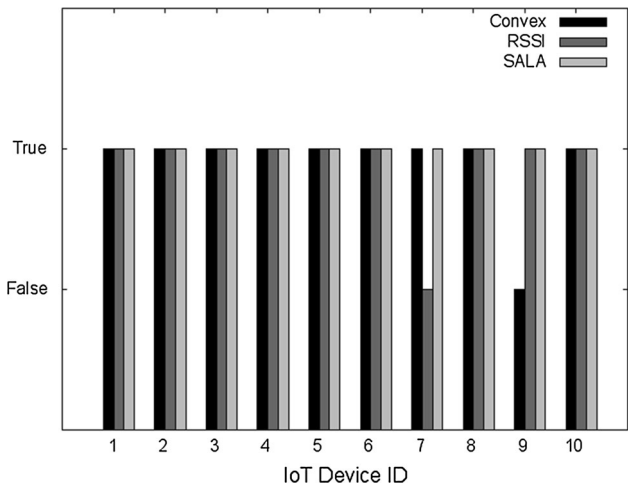


Fig. 14 Localization accuracy at apartment level. SALA can perfectly estimate the correct rooms of all the IoT devices, while Convex and RSSI make a mistake in the room estimation for the IoT devices, respectively

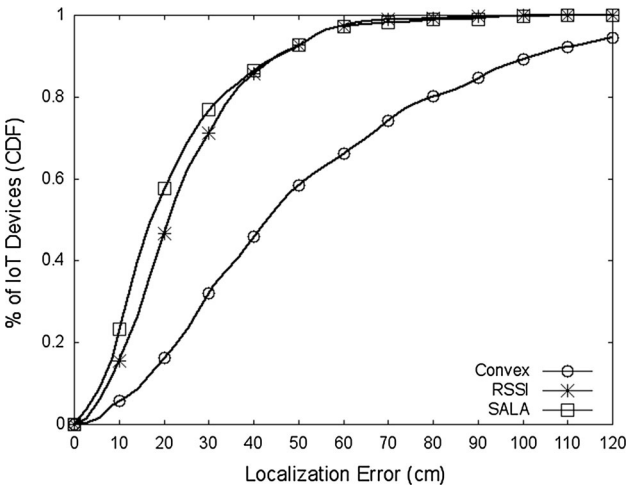


Fig. 16 CDF of localization error. SALA lets most of IoT devices have smaller localization errors than Convex and RSSI

performance over time in terms of average localization error and average localization accuracy.

Figure 17 shows the localization error over time. Note that in the figure, the localization error for each IoT device is the average of 30 measurements. In this figure, a 95 % confidence interval of a vertical value is shown for each horizontal point. It is noted that a 95 % confidence interval is used in this paper. As shown in this figure, the localization errors of all the three schemes tend to decrease over time. This is because more sample points for localization can reduce the impact of irregular communication ranges between the smartphone and each IoT device under the Rician radio channel model. That is, more sample points grant more points close to the expected communication range by the transmission power of the smartphone.

In Fig. 17, before 30 min, SALA has bigger errors than RSSI, but after 30 min, SALA has smaller errors than RSSI. This is because Power-Distance table becomes reliable with a sufficient number of sample points close to the communication range corresponding to the transmission power of the smartphone. That is why SALA outperforms RSSI.

Figure 18 shows the localization accuracy over time. Note that in the figure, the localization error for each IoT device is the average of 30 measurements. This figure demonstrates that SALA have a good localization accuracy of about 80 % after 50 min. On the other hand, RSSI and Convex have localization accuracy less than 80 % even after 50 min. Thus, the figures show that SALA has better performance after a certain point when the power-distance table discussed in Sect. 4.2.3 becomes reliable. This is because the accurate average distance μ_k in the power-distance table (as shown in Table 1) requires a certain amount of data.

5.4 Impact of motion error

In this subsection, we investigate the impact of motion error on the performance. The self-tracking of the smartphone depends on the motion sensors. As the smartphone is moving over time, the error of self-tracking can be accumulated by the motion error. This motion error is important in the localization accuracy because SALA and the other baseline schemes are performed, based on the position of the smartphone for the localization of IoT devices.

In Fig. 19, the localization error increases as the motion error increases. Below the motion error of 1.2 m, SALA shows a smaller localization error than both Convex and RSSI. However, above the motion error of 1.2 m, SALA has a greater localization error than RSSI. The error curve of SALA is rising relatively more quickly than those of RSSI and Convex. From this figure, it is observed that SALA is more sensitive in motion error than the other schemes. However, with the legacy smartphone-based self-tracking techniques [12, 32], the mean error can be bounded with 150 cm. Also, to further reduce accumulated motion error during the travel, beacons with location information can be deployed sparsely as landmarks in the target indoor place [28]. Thus, with a better self-tracking service, SALA will have a better performance.

In Fig. 20, the localization accuracy decreases as the motion error increases. Below the motion error of 0.7 m, SALA has better accuracy than both RSSI and Convex. The accuracy curve of SALA is falling more quickly than those of RSSI and Convex. From this figure, in the same way with localization error in Fig. 19, it is observed that SALA is more sensitive for smartphone's self-tracking than the other schemes.

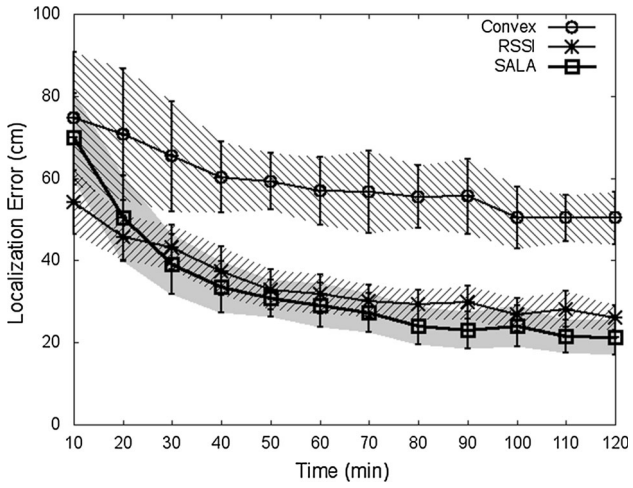


Fig. 17 Localization error over time. Since power-distance table becomes reliable after 30 min, SALA outperforms Convex and RSSI after 30 min

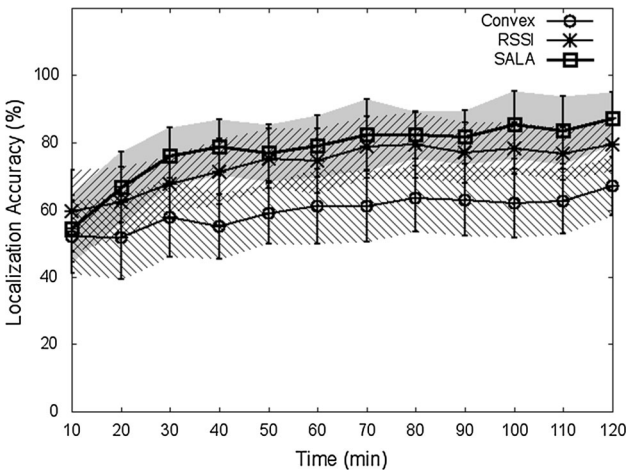


Fig. 18 Localization accuracy over time. Since power-distance table becomes reliable after 50 min, SALA has a good localization accuracy above 80 %

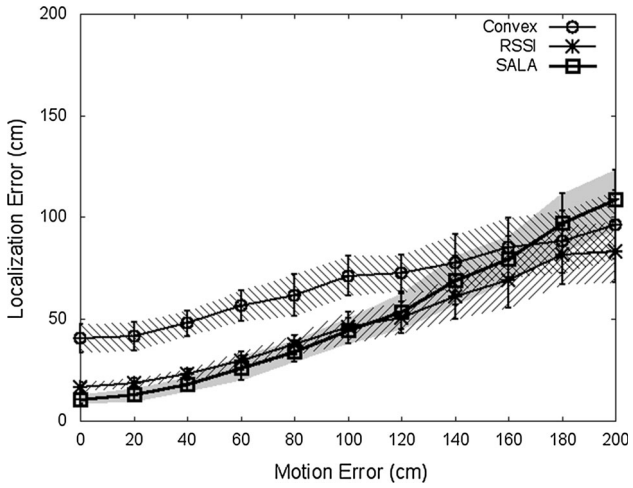


Fig. 19 Localization error over motion error. Below the motion error of 1.2 m, SALA shows smaller localization error than Convex and RSSI. SALA will have better performance with better self-tracking service in dead reckoning

5.5 Localization of passive measurement

This subsection shows the feasibility of the indoor localization of SALA through passive measurement. SALA has so far used the active measurement of beacon reply messages. However, SALA can use passive measurement for indoor localization. The passive measurement monitors radio data traffic between IoT devices and a WiFi AP in the apartment in order to collect sample points for localization. When a smartphone receives a WiFi frame from an IoT device, it stores the IoT device's MAC address (as IoT device ID), the reception signal strength (i.e., RSSI), and the location of the smartphone.

Figure 21 shows the localization error of ten IoT devices according to IoT device ID through passive measurement. SALA has better performance than the others for IoT devices 3, 4, 5, 6, 7, and 9. In Fig. 22, SALA always has smaller error than the other schemes. As expected, all schemes show better performance as time passes. This is because more sample points are available for better localization. This figure demonstrates that SALA can support the indoor localization of IoT devices through passive measurement as well as active measurement.

Overall, simulation results have shown that SALA can provide a promising indoor localization using the self-tracking functions of a smartphone.

5.6 Validation through real experiment

In this subsection, SALA is validated through real experiment based on smartphone App and Raspberry Pi devices [33]. Fig. 23 shows our SALA system architecture for the real experiment. It is mostly the same with the SALA system architecture shown in Fig. 3, but a few things have been modified so that the system can explain more about the implementation in a real environment. Because the system uses passive measurement, it is designed to perform one-way communication with IoT devices. As shown in the figure, SALA system consists of three kinds of nodes, such as smartphones, IoT devices and SALA server. Raspberry Pi 2 Model B and DW-400MINI were used as IoT devices and a smartphone of Galaxy SHV-E160s was used. The localization of IoT devices is performed through the following six steps:

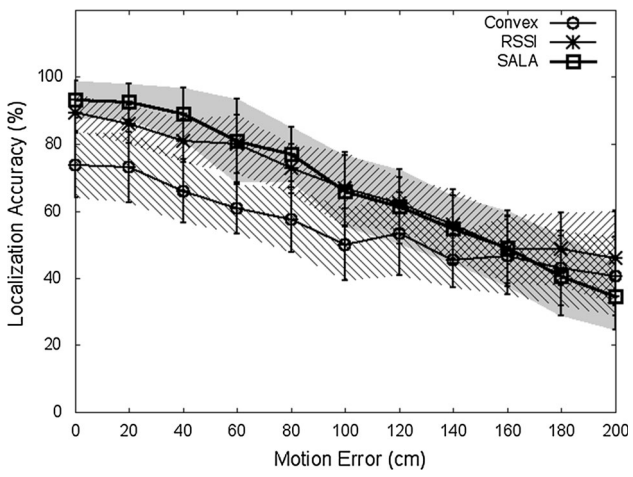


Fig. 20 Localization accuracy over motion error. Below the motion error of 0.7 m, SALA has better accuracy because SALA is more sensitive by dead reckoning than the other schemes of Convex and RSSI

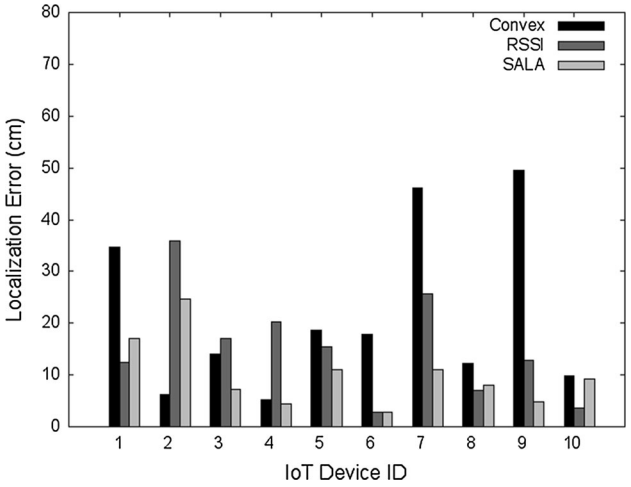


Fig. 21 Localization Error of IoT Devices on Passive Model. SALA has better performance for IoT devices 3, 4, 5, 6, 7, and 9. SALA can work well through not only active measurement, but also passive measurement

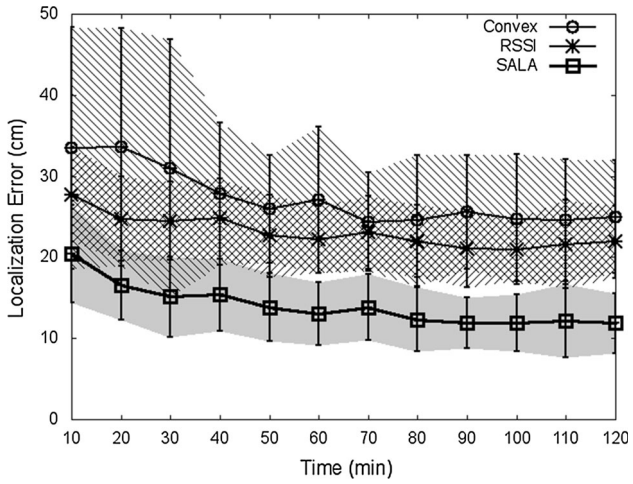


Fig. 22 Localization Error over Time on Passive Model. SALA always has smaller error than the other schemes since SALA can work well through not only active measurement, but also passive measurement

1. *IoT device detection* Each IoT device d_i keeps broadcasting a beacon message for the measurement of the RSSI w_j^i by a smartphone s_k where i is the device index, j is the time instant index, and k is the smartphone index.
2. *Dead reckoning* The smartphone s_k keeps tracking its own position p_j by performing dead reckoning [13] according to sampling time instant t_j. The dead reckoning is performed by using a 9-axis sensor, step sensor, and particle filter. The 9-axis sensor consists of gyroscrope, accelerometer, and magnetometer, with which any smartphone device is equipped. The step sensor is the modification of the linear accelerometer that can detect human steps with the value of vibration. A particle filter [13] is an algorithm that can improve the accuracy of dead reckoning, especially under narrow indoor environment like corridor. Smartphone s_k periodically calculates the power of the message (i.e., RSSI) w_j^i from the IoT device d_i, associates the power with its current position for a pair, and stores this pair into its local repository. When s_k collects a sufficient number of coordinates for d_i, s_k reports the set C_i of tuples (p_j^i, w_j^i) for d_i to SALA Server.
3. *Collection of IoT device detection trace* SALA Server collects IoT device d_i's trace data C_i that are a set of tuples (p_j^i, w_j^i) where p_j^i is a detection position for d_i, and w_j^i is the RSSI of the broadcast message from d_i.
4. *SALA* SALA Server performs SALA to compute an estimate p_i^* of the IoT device d_i's position from the trace data C_i.
5. *Position notification* When the location information l_i is updated or the new smartphone s_k is connected to SALA Server, the information l_i delivered to s_k so that the smartphone's user can see where the IoT device d_i is located.
6. *Visualization* Smartphone s_k displays the location of each IoT device d_i by using l_i, as shown in Fig. 1b.

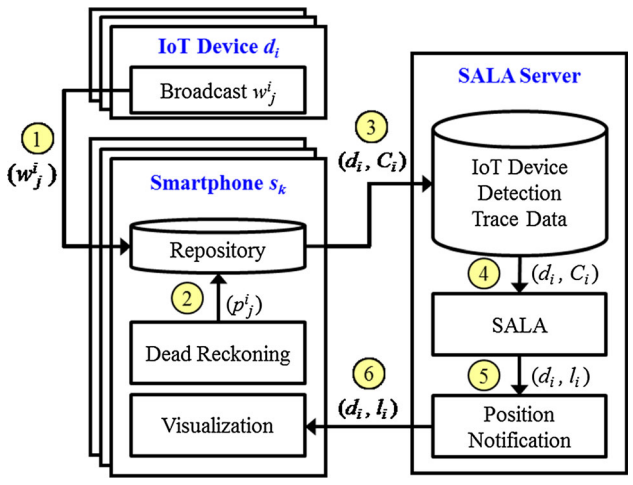


Fig. 23 SALA system architecture for real experiment. localization procedure consists of (*1*) IoT device detection, (*2*) dead reckoning, (*3*) collection of IoT device detection trace, (*4*) SALA, (*5*) position notification, and (*6*) visualization

The experiment was conducted in the office with 8.7 m × 6.6 m area. Fig. 24 shows the real experimental environment for SALA. The three IoT devices are deployed in a corner, wall, and open space, respectively. Device 1 is located at the bottom-right corner, Device 2 is located close to the bottom-middle wall, and Device 3 is located in the middle of the open space. The room shown in Fig. 25 is surrounded by many wireless devices, such as APs, and so there is some possibility of the noise higher than a common indoor place. It is assumed that at the entrance of the room, the user's smartphone sets its accurate coordinate by either the user's input or a beacon (having a position information). After this setting, the user is moving around the room, holding the smartphone with its screen upward. When the user turns on the SALA App in the smartphone, it starts to prepare for initializing sensors and WiFi. If the step sensor notifies the SALA App of the occurrence of each step or turning, it updates its location. The smartphone makes a list of RSSIs of the broadcast messages for an IoT device's ID and transfers it to SALA Server. SALA Server extracts the device detection trace, performs SALA localization, and then sends the location information back to the smartphone. At last, the smartphone shows the estimated location of IoT devices.

Figure 26 shows the localization error of the three IoT devices in Fig. 24 according to IoT device ID where about 300 sample points per IoT device are collected. Note that in

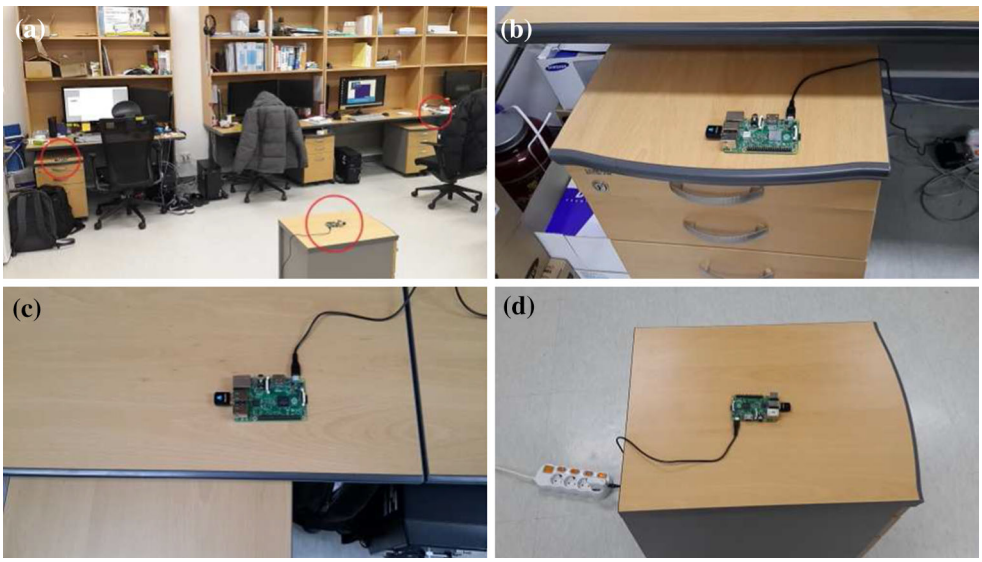


Fig. 24 Indoor IoT device deployment in office. A raspberry Pi connected to AP is used as an IoT device. **a** Three devices are located in office. **b** A nearby-corner device. **c** A nearby-wall device. **d** An open-space device

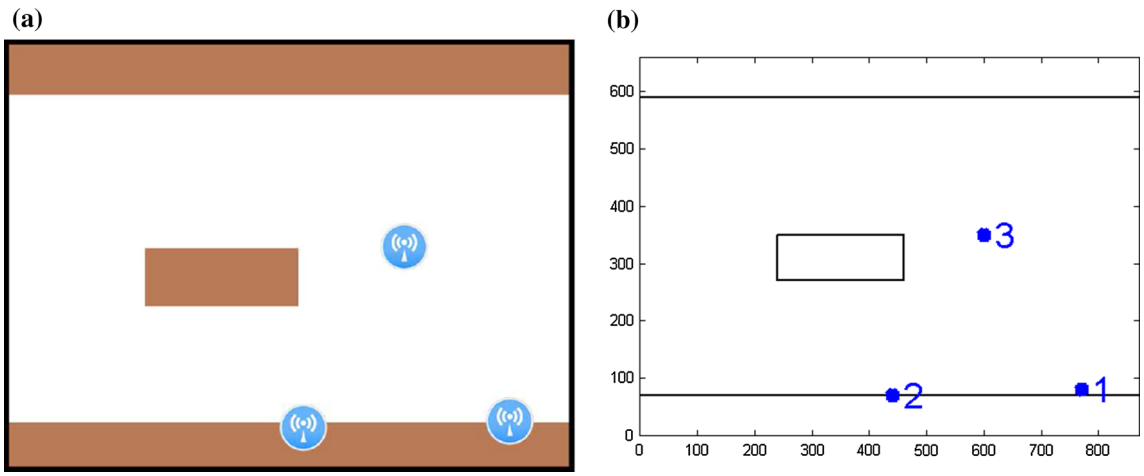


Fig. 25 IoT device deployment map. **a** The layout of the room with IoT devices. **b** The real positions of devices, such as a nearby-corner device (denoted as 1), a nearby-wall device (denoted as 2), and an open-space device (denoted as 3)

the figure, the localization error for each IoT device is the average of 30 measurements. In most cases, SALA has a smaller localization error than both Convex and RSSI. In term of the average of the localization errors for the three IoT devices, SALA outperforms both RSSI and Convex where SALA, Convex, and RSSI have the average localization error of 158.9, 259.8, and 166.4 cm, respectively. The result is similar to the simulation result but has lower accuracy. However, in the experiment, the motion error was higher than the expected one. To improve the accuracy of SALA, the accuracy of the dead reckoning for self-tracking should be improved. This improvement of dead reckoning is left as future work.

Figure 27 shows the localization error over steps. Note that in the figure, the localization error for each IoT device is the average of 30 measurements over steps. In this figure, SALA has the best performance among all three schemes. From this figure, the localization error of both SALA and RSSI is getting smaller according to steps, but that of Convex is high regardless of steps.

Therefore, through simulations and real experiments, it is shown that SALA outperforms both RSSI and Convex in terms of localization error, and so SALA is a promising indoor localization scheme for IoT devices, based on smartphones.

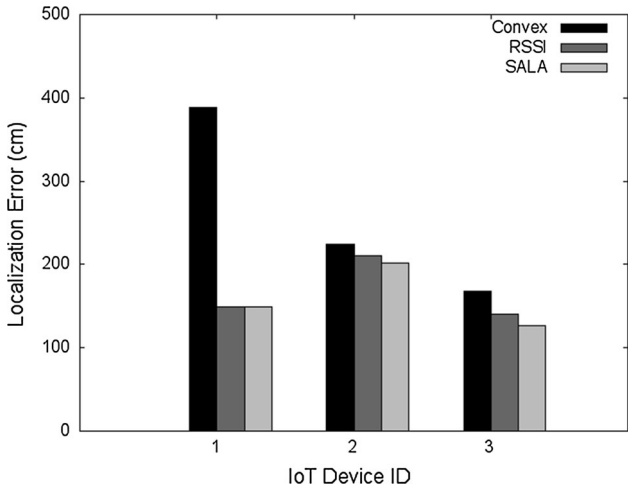


Fig. 26 Localization error of IoT devices in real experiment. In most cases, SALA has smaller localization error than Convex and RSSI

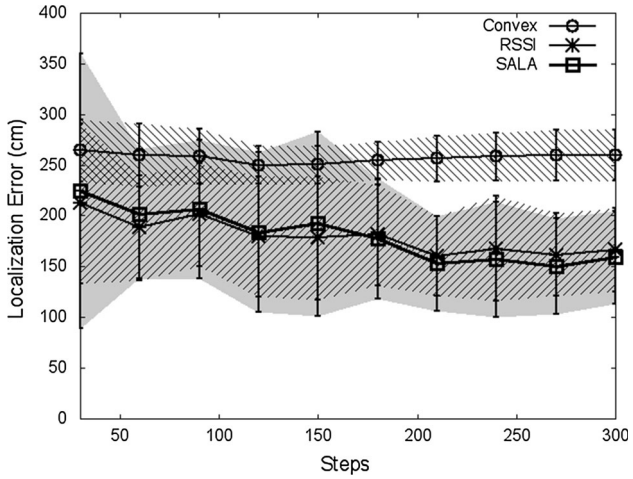


Fig. 27 Localization error over steps in real experiment. SALA outperforms convex and RSSI in terms of localization error

6 Conclusion

This paper proposes a Smartphone-Assisted Localization Algorithm called SALA to position IoT devices in indoor environments. Our SALA uses a smartphone as a mobile anchor node that has its up-to-date position during its movement at home through its motion sensors. Along with this smartphone's position, the smartphone communicates with IoT devices in a short distance to collect the proximity information. With the obtained proximity sample points, our SALA can estimate the locations of IoT devices that may be located in the middle of the apartment, closely to a wall, or close to a corner. Through simulation under a realistic radio model, it is shown that our SALA can effectively localize IoT devices in the apartment. As future work, we will deploy SALA for the indoor localization of a real, large-scale IoT network having many IoT devices, such as mall and factory. Also, we will develop a localization scheme using virtual beacons (based on WiFi fingerprints and motion sensor signatures) as landmarks to correct the localization error that is caused by the accumulated motion error.

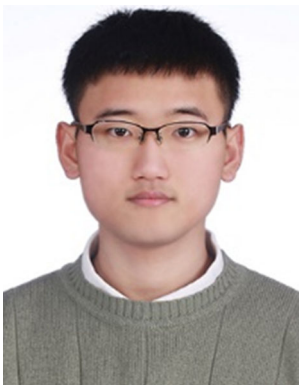
Acknowledgments This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2014006438). This work was also partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) [10041244, SmartTV 2.0 Software Platform].

References

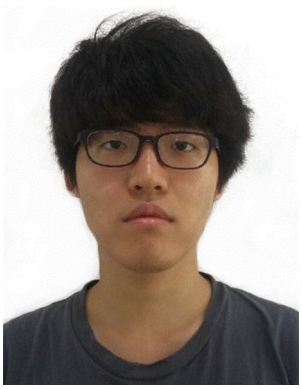
1. Cook, D. J., Youngblood, M., Heierman, III, E. O., Gopalratnam, K., Rao, S., Litvin, A., & Khawaja, F. (2003). MavHome: An agent-based smart home. In *PerCom*, Fort Worth, TX, USA, March 2003. IEEE.
2. AllJoyn Framework. AllSeen Alliance: A leading internet of things initiative. https://allseenalliance.org/framework.
3. Shala, U., & Rodriguez, A.. (2011). Indoor positioning using sensor-fusion in android devices. Technical report, 2011. http://hkr.diva-portal.org/smash/get/diva2:475619/FULLTEXT02.
4. Chintalapudi, K., Padmanabha Iyer, A., & Padmanabhan, V. N. (2010). Indoor localization without the pain. ACM Mobicom, September 2010.
5. He, X., Badiei, S., Aloi, D., & Li, J. (2013). WiFi iLocate: WiFi based indoor localization for smartphone. In *Wireless telecommunications symposium*. April 2013.
6. Jiang, J.-A., Zheng, X.-Y., Chen, Y.-F., Wang, C.-H., Chen, P.-T., Chuang, C.-L., & Chen, C.-P. (2013). A distributed RSS-based localization using a dynamic circle expanding mechanism. *IEEE Sensors Journal, 13*(10), 3754–3766.
7. Otsason, V., Varshavsky, A., LaMarca, A., & de Lara, E. (2005). Accurate GSM indoor localization. In International conference on ubiquitous computing. September 2005.
8. Liu, H., Yang, J., Sidhom, S., Wang, Y., Chen, Y., & Ye, F. (2014). Accurate WiFi based localization for smartphones using peer assistance. *IEEE Transactions on Mobile Computing*, *13*(10), 2199–2214.
9. Liu, K., Liu, X., & Li, X. (2013). *Guoguo: Enabling fine-grained indoor localization via smartphone*. In *MobiSys.* ACM, June 2013.
10. Hsu, H.-H., Peng, W.-J., Shih, T. K., Pai, T.-W., & Man, K. L. (2014). Smartphone indoor localization with accelerometer and gyroscope. In *International conference on network-based information systems*, September 2014.
11. Liu, J., Chen, R., Pei, L., Guinness, R., & Kuusniemi, H. (2012). A hybrid smartphone indoor positioning solution for mobile LBS. In *Sensors*, December 2012.
12. Li, F., Zhao, C., Ding, G., Gong, J., Liu, C., & Zhao, F. (2012). A reliable and accurate indoor localization method using phone inertial sensors. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, UbiComp '12, pages 421–430, New York, NY, USA, 2012. ACM.
13. Jun, J., Gu, Y.., Cheng, L., Sun, J., Zhu, T., & Niu, J. (2013). Social-Loc: Improving indoor localization with social sensing. In *SenSys.* ACM, November 2013.

14. Priyantha, N. B., Chakraborty, A., & Balakrishnan, H. (2000). The cricket location-support system. In *MOBICOM*. ACM, August 2000.
15. Niculescu, D., & Nath, B. (2003). Ad hoc positioning system (APS) using AOA. In *INFOCOM*. IEEE, March 2003.
16. Niculescu, Dragos, & Nath, Badri. (2003). DV based positioning in Ad hoc networks. *Telecommunication Systems*, *22*(1), 267–280.
17. Chen, H., Sezaki, K., Deng, P., & Cheung So, H. (2008). An improved DV-Hop localization algorithm for wireless sensor networks. IEEE, In *ICIEA*. June 2008.
18. Yu, N., Wan, J., Song, Q., & Wu, Y. (2006). An improved DV-hop localization algorithm in wireless sensor networks. In *ICIA*. IEEE, August 2006.
19. Chen, H., Sezaki, K., Deng, P., & Cheung So, H. (2012). Improved DV-hop localization algorithm for wireless sensor networks. In *SISY.*, IEEE, September 2012.
20. Zhong, Z., & He, T. (2009) Achieving range-free localization beyond connectivity. In *SenSys*, November 2009.
21. Zhong, Z., Zhu, T., Wang, D., & He, T. (2009). Tracking with unreliable node sequences. In *INFOCOM.*, IEEE, April 2009.
22. Feng, C., Au, W. S. A., Valaee, S., & Tan, Z. (2012). Received-signal-strength-based indoor positioning using compressive sensing. *Transactions on Mobile Computing*, *11*(12), 1983–1993.
23. Koweerawong, C., Wipusitwarakun, K., & Kaemarungsi, K. (2013). Indoor localization improvement via adaptive RSS fingerprinting database. In *ICOIN.*, IEEE, January 2013.
24. Chen, Y., Lymberopoulos, D., Liu, J., & Priyantha, B. (2012). FM-based indoor localization. In *MobiSys.*, ACM, June 2012.
25. Matic, A., Papliatseyeu, A., Osmani, V., & Mayora-Ibarra, O. (2010). Tuning to your position: FM radio based indoor localization with spontaneous recalibration. In *PerCom.*, IEEE, March 2010.
26. Ruiz-Ruiz, Antonio J., Canovas, Oscar, Rubio Muñoz, Ruben A., & Lopez de Teruel Alcolea, Pedro E. (2012). Using SIFT and WiFi signals to provide location-based services for smartphones. *LNICST*, *104*, 37–48.
27. Nirjon, S., Dickerson, R. F., Asare, P., Li, Q., Hong, D., Stankovic, J. A., Hu, P., Shen, G., & Jiang, X. (2013). Auditeur: A mobile-cloud service platform for acoustic event detection on smartphones. In *MobiSys.*, ACM, June 2013.
28. Martin, P., Ho, B.-J., Grupen, N., Muñoz, S., & Srivastava, M. (2014). An iBeacon primer for indoor localization. In *BuildSys.*, ACM, November 2014.
29. Zhou, G., He, T., Krishnamurthy, S., & Stankovic, J. A. (2006). Models and solutions for radio irregularity in wireless sensor networks. *ACM Transactions on Sensor Networks*, *2*(2), 221–262.
30. OMNeT++. Discrete Event Simulator for Networks. https://omnetpp.org/.
31. Hyytiä, E., & Virtamo, J. (2007). Random waypoint mobility model in cellular networks. *Wireless Networks*, *13*(2), 177–188.
32. Chung, J., Donahoe, M., Schmandt, C., Kim, I.-J., Razavai, P., & Wiseman, M. (2011). Indoor location sensing using geo-magnetism. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pp.141–154, New York, NY, USA, 2011. ACM.
33. Raspberry Pi. IoT Devices. https://www.raspberrypi.org/.

Jaehoon (Paul) Jeong is an assistant professor in the Department of Software at Sungkyunkwan University in Korea. He received his Ph.D. degree from the Department of Computer Science and Engineering at the University of Minnesota in 2009. He received his B.S. degree from the Department of Information Engineering at Sungkyunkwan University and his M.S. degree from the School of Computer Science and Engineering at Seoul National University in Korea, in 1999 and 2001, respectively. His research areas are vehicular networks, cyber-physical systems, Internet of things, wireless sensor networks, and mobile ad hoc networks. His two data forwarding schemes (called TBD and TSF) for vehicular networks were selected as spotlight papers in IEEE Transactions on Parallel and Distributed Systems in 2011 and in IEEE Transactions on Mobile Computing in 2012, respectively. Dr. Jeong is a member of ACM, IEEE and the IEEE Computer Society.

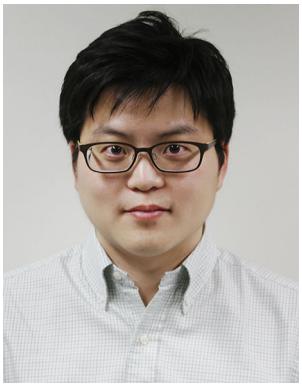


Solchan Yeon is an undergraduate student in the Department of Computer Science at Kookmin University in Korea. His research areas are wireless sensor networks and mobile ad hoc networks. Mr. Yeon is a member of ACM and IEEE.



Taemoon Kim is a graduate student in the Department of Software Platform at Sunkyunkwan University in Korea. He received his B.S. degree from the Department of Software at Sungkyunkwan University in 2015. His research areas are cyber-physical systems, Internet of things, and problem solving.

Hyunsoo Lee is an undergraduate student in the Department of Software at Sungkyunkwan University in Korea. His research interests include cyber-physical systems and artificial intelligence.



Song Min Kim is an assistant professor in the Department of Computer Science at George Mason University. He received his Ph.D. from the Department of Computer Science and Engineering at the University of Minnesota in 2016. He received his M.E. and B.E. degrees from the Department of Electrical and Computer Engineering at Korea University in 2009 and 2007, respectively. His research interests include wireless and low-power embedded networks, mobile computing, Internet of things, and cyber-physical systems.

Sang-Chul Kim is an associate professor in the School of Computer Science at Kookmin University, Seoul, Republic of Korea, since March 2006. He received the Ph.D. degree in the department of Electrical and Computer Engineering from the Oklahoma State University in 2005. He was a System Engineer in Samsung SDS and Aerospace from 1994 to 1999 and was an Instructor in the school of Computer Science at the Changwon National University from 1998 to 1999. His research is in the areas of mobile and ad hoc communication, Internet of things, and wireless sensor networks. He served as TPC Vice Co-Chairs of IEEE ICUFN 2015, Local Chair of IEEE VNC 2012, TPC member of IEEE ICUFN 2011, and Local Arrangement Co-Chairs of IEEE ICUFNs 2012, 2013, and 2014. He is a life member of the Korean Society for Internet Information (KSII) and Korean Institute of Communication and Information Sciences (KICS). He is an Editor of the KSII Transactions on Internet and Information Systems (TIIS). In April 2015, he was nominated as one of the outstanding scholars in the book of "Marquis Who's Who in the World 2016".