Beacon-based Indoor Fire Evacuation System using Augmented Reality and Machine Learning

Hwawon Lee
School of Software
Soongsil University
Seoul, Korea
andylhw12@soongsil.ac.kr

Dohyun Chung
Industrial Security
Chungang University
Seoul, Korea
sosvast@cau.ac.kr

Sungmin Kim
Computer Engineering
Kangwon National University
Chuncheon, Korea
aliveksm@kangwon.ac.kr

Jiwon Lim
Computer Engineering
Kwangwoon University
Seoul, Korea
senta2006@kw.ac.kr

Yoonha Bahng
Computer Engineering
Chungang University
Seoul, Korea
tlol91@cau.ac.kr

Suhyun Park
Computer Engineering
Paicai University
Seoul, Korea
2061013@pcu.ac.kr

***Abstract*—The inefficiency of fire evacuation has been the issue since the present evacuation method is unsuitable for complex buildings in mass society. In order to improve the evacuation system, This paper aims at three main components: high accuracy of indoor localization, real-time optimized route algorithm, and user-friendly augmented reality (AR) navigation. First of all, Kalman Filter and deep learning models were utilized to estimate the user's location accurately. Second, for dynamic individual fire evacuation path decisions, an optimized route recommendation based on Q-learning was designed, which considered the components of buildings and disaster information. The optimal evacuation algorithm prevents congestion by guiding each user to several exits. Lastly, Augmented Reality and a 2D map with heading orientation derive the navigation system without any confusion. The proposed system offers the safest path based on accurate location with a user-friendly visual supplement.**

***Index Terms*—Bluetooth Low Energy, Deep Learning, Augmented Reality, iBeacon, Indoor Localization, Real-time communication, Reinforcement Learning**

I. Introduction

From 2017 to 2019, 106,700 multifamily residential building fires were reported in the United States, resulting in 400 deaths, and 3,875 injuries [1]. When a fire occurs in a large and complex building, people who do not know the building's structure would be exposed to a danger. It will be difficult for them to be aware of their current location, as well as the exit route and the location of fire hazards in a panic situation. The inefficiency of the current evacuation method could be fatal when evacuating fire victims safely from the fire [2]. To handle such problems, safe and efficient evacuation system has been studied.

Three components of these studies have been categorized into indoor localization, optimized route algorithm and navigation. Commonly used indoor localization method is using wireless electronic devices that utilize the Wi-Fi localization method. The Wi-Fi localization to approach indicates the nearest Access Points (AP)'s location to the user [3] but not the exact user's location. Some studies used the Pedestrian Dead-Reckoning (PDR) [4], [5] method; however, since the situations during emergencies are different, the result of notifying the location by footstep was inaccurate. Optimized route algorithms were conducted with Dijkstra's algorithm and A* algorithm [6], [7] [8] adopted for the fire evacuation situation. These approaches place emphasis on calculating the shortest distance or time to find optimized route, and they are prior factors for selecting path. However, the error of these algorithms is lack of consideration that can influence safe and quick evacuation. The degree of congestion did not take into account and updated in real-time. Studies of navigation system have been conducted two-dimensional (2D) based navigation and vision based navigation [6]. In case of fire evacuation, 2D-based navigation has its limit on being dependent on angle and detecting the actual escaping route in complex building. Smoke and collapse from fire give rise to malfunction of recognizing the exact location. These systems cause evacuee not to notice various hazard, eventually put them in danger. This paper proposes are categorized in indoor localization, algorithm and guide visualization. For Indoor localization, we utilized the mobile device to catch the beacon signal, the Received Signal Strength Indicator (RSSI). Utilizing the Kalman Filter to reduce the fluctuation of the RSSI and the Deep Neural Network (DNN) model improved the accuracy of indoor localization. DNN model was inserted in the mobile. Mobile derives the estimated location of user. The safest way to guide the evacuees, Q-learning algorithm gives the optimized path and also considered the congestion to avoid people being crushed. In the guide visualization, a 2D map displays the current location of the user and congested areas to avoid and direct people with a 3-dimensional (3D)

navigator. Using both of these supplements, derives a user-friendly instruction. The main contributions of this article are summarized as follows:

- This paper proposes high accuracy indoor localization system by using beacons. The RSSI values of the beacon signals are filtered with Kalman Filter, and to calculate the distance of the receiver and the transmitters. Machine learning models were used for locating the user.
- For dynamic individual fire evacuation path, q-learning Algorithm derives route consider the hazards and the components of buildings. The optimal evacuation algorithm avoids congestion by using multiple exits. Finally, the optimized route will be sent to each user in real-time.
- The solution of a sight-blocking situation, user-friendly AR technology was adopted. In the 2D map the congestion areas and the users location are viewed and the users will be guide With a 3D Augmented Reality (AR) navigation.

II. Related Work

In this section, we will describe the technology that the other articles used for evacuation system.

A. Indoor localization

In previous research [9], Kalman Filter is used to produce optimal indoor localization. Fixed receivers obtain transmitted signals and calculate Received Signal Strength Indicator(RSSI) values. Within the receiver side, to attain better indoor localization accuracy, the Kalman filter is employed to stabilize the RSSI values. However, this study cannot be used to localize moving objects because the Kalman filter would attempt to stabilize weakening RSSI as an object moves farther away from a transmitter.

KF-SVR is introduced to deal with moving objects, which fits RSSI distribution to a more precise and accurate model [10]. Together, Support Vector Regression (SVR) simulate the correlation between RSSI and distance while Kalman Filter stabilizes varying RSSI signals. By using KF-SVR, this research improves error reduction by 40% / 20% when compared to using the Path-Loss model of the Alt-Beacon open-sourced library [11] and the Non-linear Path-loss model, respectively. However, it is still limited by the requirement that the object moves at a constant speed. Additionally, SVR model is vulnerable to surrounding obstacles such as wires and network signals. As a result, the distance estimation of the SVR model do not work well in fire disaster, since people do not move in regular speed in urgent situation.

To estimate positions of evacuees and construct the evacuation routes, beacons are placed in a row on the ceilings of corridors [12]. The intervals of the installed beacons are 10m or 12.5m. The beacon closest to each evacuees is detected with RSSI value and defined as the position of the evacuee. Other beacons are set as nodes of the path finding algorithm and the evacuation route from the current position to emergency exits is designed. Unfortunately, this research is not able to cover various indoor architecture. Since the interval distance between each beacons are more than 10m, it leads to serious errors in estimating the user position in a wide area or place with barriers.

B. Optimized Route Algorithm

The shortest path has typically been the primary consideration when designing evacuation path-finding algorithms [8]. The Dijkstra algorithm is applied to identify the most secure path-planning method at each point in an evacuation. Another crucial aspect of emergency crowd control is directing people away from danger along the safest path. As a result, simply selecting the quickest path does not guarantee a safe escape. Dijkstra's method is therefore solved for every exit node to find the path with the lowest cost. Despite the fact that this study analyzed all available exits, the algorithm only chose the lowest cost evacuation path. In this situation, the algorithm cannot stop everyone from trying to go through the same exit. As a result, choosing the shortest route does not result in the fastest evacuation.

The A* search algorithm has been used for the evacuation system [13]. A* algorithm is relatively faster than the Dijkstra algorithm for calculating the route. A* algorithm provides the shortest path to evacuation, and the alternative plan is to implement another route with weighted values. However, the disadvantage of the A* algorithm is heuristic, it only considers the shortest path to the exit, and the algorithm has a hard time adopting the new situation in such a short time.

C. Navigation System

Three-dimensional (3D) Navigation was adopted without an arrow, only using pictograms [14]. This article used a pictogram and showed the remaining distance and temperature of the next checkpoint. However, when people have isolated the building filled with smoke, a pictogram may not help as expected. Also, the situation going backward is not provided, which could bring a misunderstanding and confusion in instructing the users.

The route on the floor has been marked by AR along with showing the remaining distance amount, making a user-friendly interface [4]. In addition, 2D map was also provided, so the users can see the data on the remaining distance. However, it is an image-based system so in order to notice the current location, it takes time to take a photo of the room number and communicate with the server to confirm the room number. This paper also has a limitation when the room is full of smoke.

III. Methodology

The proposed system and service model is presented in Fig. 1. This paper monitors the environment by using Raspberry Pi and ESP32. With the help of ESP32 and Raspberry Pi, they gather temperature and humidity data from sensors to detect fire. Raspberry Pi also receives signals from beacons and keeps track of the available beacons. The server notifies users using Firebase Cloud Messaging (FCM) once it receives a report that a fire has broken out. The user's smartphone starts to

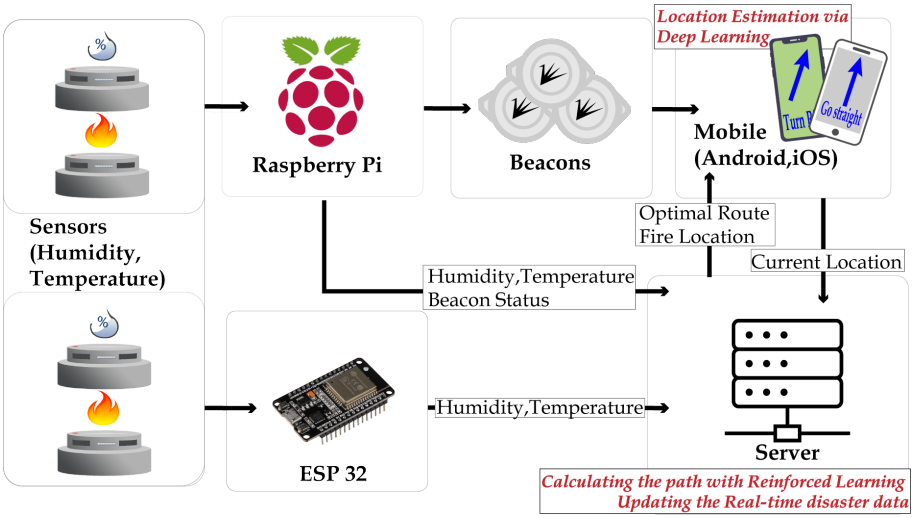


Fig. 1. Proposed system and service model.

catch the beacon signal and calculate the RSSI [15]. Kalman Filter was used to reduce RSSI fluctuation before utilizing it for localization [16]. The DNN model is then used to estimate the current position and send it to the server. Based on the location of each user, the server uses Q-learning to determine the best evacuation route. Sensor-based disaster context and congestion are taken into account in real-time when computing the path. The optimal route for each user is transmitted to the user's smartphone. Then the escape path is guided by AR. AR was created to be compatible with both iOS and Android platforms and assist evacuees in escaping safety in a simple and user-friendly way [17].

A. Indoor localization

The mobile device collects RSSI values of 22 beacon signals to estimate where the user is located.

$$\mathbf{RSSI}_{I,C} = (RSSI_1, RSSI_2, \ldots, RSSI_N) \tag{1}$$

However, signal fluctuation may occur when collecting the RSSI values [18]. This can be solved by using the Kalman filter. The Kalman filter estimates future state based to the current state, and it stabilizes the fluctuating RSSI.

As the mobile measure RSSI, mobile performs filtering with Kalman Filter. It predicts a filtered value based on the previous estimated one. Following equation expresses prediction stage.

$$\widetilde{RSSI}_{\tau'} = \mathrm{F}\widetilde{RSSI}_{\tau} + \omega_{\tau} \tag{2}$$

$\widetilde{RSSI}_{\tau}$ is the filtered RSSI value of the τ-th beacon. The prediction of RSSI is the system vector multiplied by the current filtered RSSI, lastly adding the system noise. After the prediction phase, Kalman Gain and new measured RSSI will update the previous estimation of RSSI as an output. The output will used for training multi-classification model. Following equation is the update phase of Kalman Filter [10].

- Update Current estimation

$$\widetilde{RSSI}_{\tau} = \widetilde{RSSI}_{\tau}^{-} + K_{\tau}(Z_{\tau} - H_{\tau}\widetilde{RSSI}_{\tau}^{-}) \tag{3}$$

TABLE I
VARIABLE AND NOTATIONS IN KALMAN FILTER

Symbol	Meaning	Symbol	Meaning
F	System Vector	Z	Measured RSSI
$\widetilde{RSSI}$	Filtered RSS	I	Interval of advertising beacon
K	Kalman gain	N	Number of beacons
H	Measure vector	W	System noise
P	Error covariance	R	Measure noise vector
C	Cell	**RSSI**	RSSI vector

- Update current error covariance

$$P_{\tau}^{+} = (I - K_{\tau}\mathrm{H}_{\tau})P_{\tau}^{-} \tag{4}$$

- Update Kalman Gain

$$K_{\tau} = P_{\tau}^{-}H_{\tau}^{T}(R_{\tau} + H_{\tau}P_{\tau}^{-}H_{\tau}^{T})^{-1} \tag{5}$$

Unfortunately, there is a limitation with the Kalman filter. Since the Kalman filter is suitable for an object moving linearly, a delay occurs when moving non-linearly. To solve this problem, based on the variation of RSSI value in a stationary status, the Kalman filter is initialized when a certain amount is over the threshold.

Owing to the limitations of the Kalman filter as mentioned earlier, an error occurred when determining the user's location moving in real time. For every cell in K-SW building, the adjacent cells by the direction are defined in advance. If there is a wall or there is no evacuation route, it's defined as "unkwown". When the user moves, the Kalman filter is initialized based on the received sensor values. The geo-magnetic sensor of the mobile device calculates the azimuth in order to determine which direction among north, east, south, west the user faces. Only when the user travels in the direction the user is looking, the current location is updated to the adjacent cell. If the model output does not correspond to the adjacent cells but matches the user direction, it is updated to the nearest cell among the adjacent cells. When the user

Algorithm 1: Estimating location process

Input: Previous location, Current location, Heading positions = [N, S, W, E]
Output: Estimated location
if *Previous location = NULL* **then**
  return Current location
end
else
  adjacent cell ← adjacent cell of Previous location
  if *Current location ∈ adjacent cell* **then**
    if *Heading positions ∈ Heading positions of Previous location* **then**
      return Current location
    end
    else
      return Previous location
    end
  end
  else
    if *Current location ∈ Heading positions of Previous location* **then**
      if *adjacent cell of Previous location ∈ Heading positions* **then**
        return Heading positions of Previous location
      end
      else
        return Previous location
      end
    end
  end
end

locates at A07, south adjacent cell is A06, east adjacent cell is unknown, west adjacent cell is A10 and north adjacent cell is A12, S07. If the movement is detected, the current cell changes to A06 if the user is looking south. Algorithm 1 describes how to update the location of the user based on the direction.

DNN Regression model estimates the distance with the received RSSI value [19]. Input of the model is the RSSI value and the the shape of the metrix is [None, 1]. In 3 hidden layers the ReLU function was stacked. The output of the model derive the estimated distance as a [None, 1] format. In the training phase, it captures the exponential relation between the distance and the RSSI values. After the model was trained, it is installed in the mobile device, and the device puts the RSSI value as an input, and the model returns to an estimated distance.

With the filtered RSSI values of beacon signals, Machine Learning models such as Regression models and Classification models are trained to estimate the user's location. In this paper, each of experiments are performed as the DNN Regression model and the DNN-based multi-classification. DNN Regression model estimates the distance with the received RSSI value. In the training phase, DNN regression model captures

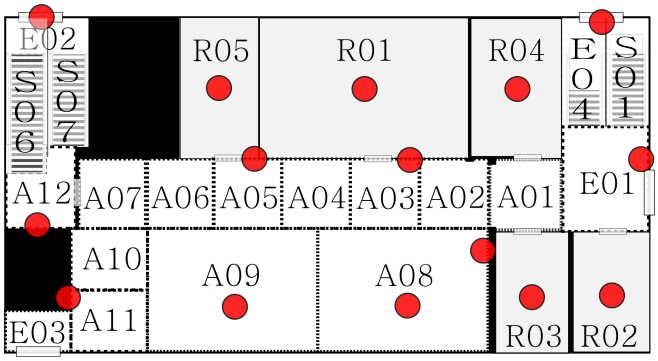


Fig. 2. Blueprint of K-SW building

the exponential relation between distance and RSSI values. When the training phase is over, the user puts the RSSI signal value as the input value, and the model returns the estimated distance to the user.

In order to confirm the characteristics of the falling RSSI exponentially, we measured the signal in units of every 0.1 meter. The environment was restricted as follows:

- Place a beacon and mobile devices in horizon
- Collect the data for 5 minutes
- Keep LOS situation, with no other obstacles between mobile devices and a beacon
- Cell phone was not held by person, placed on the chair.

The results of the experiment will be explained in section IV.

DNN classification model predicts the cell label where user is located by collecting beacons signals. In the precede model, the input shape is [None, 22] format which is RSSI vector. The number of hidden layers is 3. ReLU function is the activation function in the hidden layer [20]. He initialization was conducted to initialize the ReLU function. Furthermore, we accelerated convergence of stochastic gradient descent and improve the conditioning of the optimization problem at each hidden layer, stacking a weight normalization layer that re-parameters the weights [21]. Finally, Softmax activation function, which is widely adopted multi-classification, was utilized, and the probability for 31 classes is derived in the [None, 31] matrix form. For using classification models, K-SW Square building was divided into cells and the RSSI values were measured at each cell. For efficient localization considering the structure of K-SW, different type of rectangles were chosen for division. As shown in Fig. 2, we divide building into 31 sections by using cell type as TABLE II. The places where the user can move in multiple directions need more fine-grained localization. Those places are divided into small-sized cells, 2.5m * 2.5m. For example, the user located in A05 can evacuate via A04 or A06 or A09. In contrast, there are some places which is larger than 2.5m * 2.5m. These area do not influence the exit path or area which is not divided as others. For instance, A08 is not divided into 2.5m * 2.5m, and usually not considered as part of escape route.

The RSSI values measured on each cells are preprocessed to eliminate the outlier values. The preprocessed RSSI data

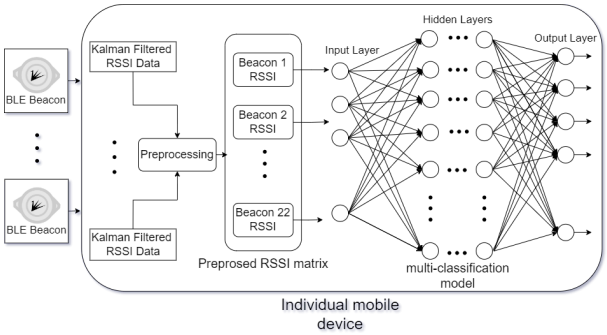


Fig. 3. DNN overview for mobile device

TABLE II
RELATIONSHIP BETWEEN CELL TYPE AND ITS DESCRIPTION

Cell type	Description
A	Normal area
R	Room
S	Stair area
E	Exit area

are provided as input of the regression model, and its output is the cell label of the user's location.

In this experiment setup, DNN classification model is trained to predict the user's location with preprossed RSSI values. As -90dB was too weak to receive, the RSSI value more than -90dB was adopted as vaild value and the rest of the values are categorized as invalid values. The following invalid values are replaced to -200dB.

Among the preprocessed values the top 4 values were adopted as valid values. Both valid and invalid values are used as features for every 22 beacons. While collecting from the mobile devices, the values were converted as a TensorFlow lite model.

As shown in Fig. 3, in the actual experiment, the model was ported to Tensorflow Lite for the mobile device, because it supports both Android OS and iPhone OS. The RSSI values from each beacon are collected and filtered with Kalman filter. The filtered RSSI values can be changed to an invalid signal based on the above conditions. If there is a loss value among the continuously received values in real time, it is adjusted to a previous value. As the DNN Regression model is vulerable to environmental changes, it is not suitable for fire evacuation systems which has lots of unexpected situations. As a result, the DNN Classification model was selected in this paper. Experimental results of both models are described in section IV.

B. Evacuation pathfinding algorithm

After estimating the user's location, the mobile application submits their location to the server and receives an evacuation path. The Q-learning algorithm, a form of Reinforcement learning (RL), is used by the server to provide the evacuation route. A set of states S and a set of actions A in our proposed Q-learning are defined in Equation 6.

$$\begin{aligned} S &= \{cell_1, cell_2, cell_3, \ldots, cell_N\} \\ A &= \{A_1, A_2, A_3, \ldots, A_N\} \\ A_i &= \{a_i \mid a_i \text{ is adjacent cell of } cell_i\} \end{aligned} \tag{6}$$

Where N is the number of cells. By using action-value function called Q-table, Q-learning algorithm finds the optimal cell for the following action. For finding the optimal policy, the Q-

Algorithm 2: Proposed Q-Learning for Evacuation Pathfinding

Input: Cells topology, Number of episodes N_{epi}, Destination cell
Output: Q-table for Destinaion cell
for *iteration* $\leftarrow 1$ **to** N_{epi} **do**
  currenct cell $\leftarrow$ random cell
  $epsilon \leftarrow 1 -$ iteration $/ N_{epi}$
  while *current cell* $\neq$ *Destination cell* **do**
    pick next cell by ϵ - greedy approach
    Update Q(s, a) by using Equation 7.
    currenct cell $\leftarrow$ next cell
  end
end

Algorithm 3: Calculates path for Evacuation

Input: Cells topology, Starting cell, Destination cell
Output: Path from Starting cell to Destination cell
currenct cell $\leftarrow$ Starting cell
path $\leftarrow$ [Starting cell]
while *current cell* $\neq$ *Destination cell* **do**
  pick next cell by greedy approach
  currenct cell $\leftarrow$ next cell
  append current cell to path
end
return path

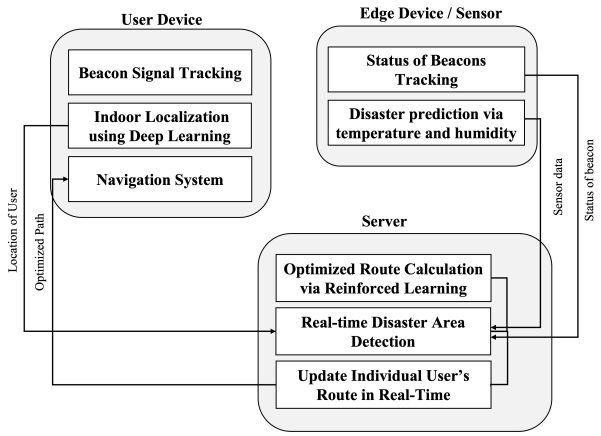


Fig. 4. Proposed system functional model.

values of state s and action a are updated as the Equation 7.

$$Q(s,a) = (1-\alpha)Q(s,a)+\alpha[R(s,a)+\gamma \times \max Q(s',a')] \quad (7)$$

Where α is the learning rate, γ is the discount factor, $R(s,a)$ is the reward for taking action a in the state s, and $\max Q(s',a')$ is the maximum predicted reward for next cell. $R(s,a)$ changes during the evacuation process, to consider the fire and congested conditions indicates in Fig. 4. Raspberry pies and the ESP32 use their sensors to collect temperature and humidity data and transmit it to the server. When temperature or humidity passes the threshold, the server assigns fire reward r_f to the cell, where the raspberry pie or ESP32 is mounted and assigns fire adjacent reward r_a to the surrounding cells. Additionally, since high temperatures can damage beacon hardware, if the raspberry pie discovers that the beacon is no longer transmitting signals, it sends a status message to the server inferring that the beacon is in a fire. Server assigns fire reward r_f to the cells that are nearby the dead beacon and assign adjacent cell reward r_a to the surrounding cells. The server notifies the number of people through the user's information. If there are more than 5 people in a cell, the server assigns the congested reward r_c to that place. r_f forces to avoid using fire cells if at all possible and r_a and r_c reduce the possibility of using fire adjacent cells and congested cells. Algorithm 2 describes how the proposed Q-learning was trained. For each iteration, a random cell was selected for a starting cell. After the training phase, every cell gets a path to the final destination. During the training phase, ϵ was initialized by 1 and gradually reduced by each episode on Equation 8.

$$\epsilon = 1 - \text{iteration count}/\#\text{ of episodes} \quad (8)$$

Next cell will be chosen randomly with a probability of ϵ and greedily in probability of 1 - ϵ. Gradually reducing the ϵ could make the Q-learning explore possible path lists and finds the optimal path. After the training phase, epsilon becomes zero and works like a greedy algorithm. Algorithm 3 takes $O(1)$ time complexity for each step. After the server yields a path for every available exit, their efficiency will be compared by using Equation 9.

$$\begin{aligned}\text{Efficiency of path} = \text{Pivot} - (\text{Length of path} \\ +F_f \times \#\text{ of fire cell in path } + F_a \times \#\text{ of fire adjacent cell} \\ +F_c \times \#\text{ of congested cell in path})\end{aligned} \quad (9)$$

Respectively F_f, F_a, F_c are the Factors that fire weight, fire adjacent, and congested cell, respectively. Scaling factors are all positive integers. Pivot is a large positive integer to ensure the efficiency of the path will be a positive value. Then the two most effective routes will be considered candidates, and the ratio of each path will be used to choose the evacuation route as described in Equation 10.

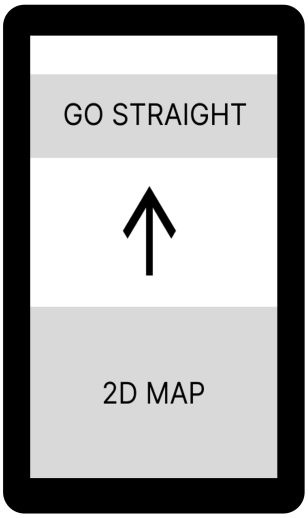


Fig. 5. Wireframe of AR

$$\begin{gathered}\text{M: Most efficient path} \\ \text{S: Second most efficient path} \\ \text{EM: Efficiency of M} \\ \text{ES: Efficiency of S} \\ \text{Path} = \begin{cases} \text{M} & \text{with probablity } \dfrac{EM}{EM+ES} \\ \text{S} & \text{with probablity } \dfrac{ES}{EM+ES} \end{cases}\end{gathered} \quad (10)$$

By split user, Q-learning can achieve optimal evacuation time and fastest evacuation path.

C. Guidance with Augmented Reality

We will use the ARKit and ARCore framework for our AR system. This system always place the 3D arrow on top to notice the direction of the next cell. In order to place a 3D arrow navigation in front of a camera in AR world, the camera orientation and position were changed to absolute orientation and position. If the camera coordinates are P(c) = (xc, yc, zc) and the absolute coordinates are P(a) = (xa, ya, za). the center of each coordinate ray is converted to a 4*4 matrix. By adding two matrices, camera coordinate can be converted to absolute coordinate. To give directions to users, we built a 2D coordinate system based on absolute orientation and KSW map position. The arctangent function was used to find the absolute degree from the user's current location to the next path location. The equations of the finding the absolute degree is below in Equation 11.

$$\begin{aligned}atan2(y,x) = arctan(\frac{y}{x})[x \neq 0] + (1 - 2[y < 0]) \\ (\pi[x < 0] + \frac{\pi}{2}[x = 0]) + \text{undefined}[x = 0 \wedge y = 0]\end{aligned} \quad (11)$$

We built a 2D coordinate system based on absolute orientation and map position of K-SW Sqaure. By using the camera

TABLE III
COLOR OF CELL AND ITS MEANING

Color of Cell	Cell Status
White	Normal area cell
Green	Evacuation path cell
Yellow	Congestion cell
Orange	Risky area cell
Red	Fire area cell

TABLE IV
CELL SHAPE AND ITS MEANING

Cell shape	Description
Circle	Current user's location
Arrow	Current user's heading orientation

coordinate converted to absolute coordinate, user can obtain the degree of next path location. In order to guide the next path using AR, the 3D object was floated at the location of the next path. User-friendly system was built using pictograms when floating 3D objects. The status of the cell is shown at TABLE III, Annotation indicating the user's status is shown at TABLE IV.

Fig. 5 is the wireframe of the AR navigation application. It shows how the AR-2D based navigation application is operates. The screen is divided into three parts. AR navigation exists throughout the screen. The direction and distance from the current position to the next position are calculated to show the arrow and the red sphere pointing the checkpoint. There is a directive at the top of the screen. It provides instructions such as GO STRAIGHT, GO BACK, TURN LEFT, TURN RIGHT, GO DOWNSTAIR, GO UPSTAIR, and DANGER depending on the user's situation. This directive calculates the difference between the angle currently viewed by the user and the angle of the arrow and shows on the screen. 2D Map is placed at the bottom of the screen. Both the user's location and the situation of the building appear. Through the auto scroll function, the user can notice the current location and surrounding situation, and the manual scroll function can get the information of the desired location.

The 2D map represents the underground, first and second floor of KSW building. The 2D map is divided in the same way as the cells divided by indoor localization. In the 2D map, each cell state, the location of the user, and the direction viewed by the user are shown.

IV. Evaluation

In this section, we will evaluate about experiments had done. In Section IV-A, Kalman-Filtering that smoothing out measured RSSI values is described. Errors incurred by Regression model and location estimation accuracy of Classification model is also depicted in section IV-A. In Section IV-B, Q-learning parameters and the evacuation simulation are described.

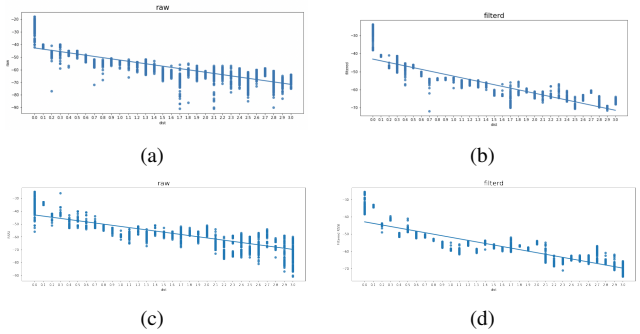


Fig. 6. Kalman Filter experiments: (a) Kalman-Filtered data of iOS, (b) Raw data of iOS, (c) Kalman-Filtered data of Android OS, (d) RAW data of Android OS

A. Indoor localization

To locate the user, the RSSI of the beacon signals were collected.

Kalman Filter was used to obtain stabilized RSSI values. Fig. 6(a) and Fig.6(c) show the distribution of the raw RSSI values over distance of Apple iPhone 12 and Samsung Galaxy A30, respectively. The distribution of filtered RSSI values over distance of iOS and Android is described in Fig. 6(b) and 6(d), respectively.

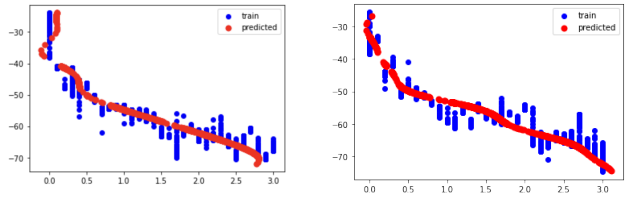
As described in the Fig. 6(a) and 6(c), some of the raw RSSI values were shown in similar distribution, despite of being measured in a different distances. In contrast, as shown in Fig. 6(b) and 6(d), the filtered RSSI values show more accurate distributions over than the Kalman Filtered distance outlier values.

Support Vector Regression(SVR) model is commonly used to estimate the distance between transmitter and receiver by using RSSI values in previous research [10]. However, the SVR model produced quite different results when the environment change. Fig. 7(a) and 7(b) show the respective distances estimated by the SVR models on ios and Android , . Kalman filtered RSSI values were given as an input of the SVR model. The RSSI values were measured in the following conditions:

- Place a beacon and mobile devices in horizon
- Collect the data for 5 minutes
- Keep LOS situation, with no other obstacles between mobile devices and a beacon
- Cell phone was not held by person, placed on the chair

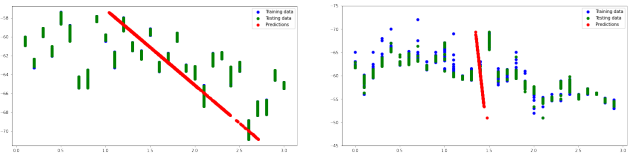
As shown in Fig. 7(a) and 7(b), the distances predicted by the SVR model show ideal distributions with exponential reduction.

However, the SVR model is vulnerable to environment changes. Further experiments were conducted to reflect real-world situation. In the further experiments, beacons were installed on the walls of the building. The beacons and mobile devices were not in horizontal alignment and were in NLoS status with lots of obstacles. Fig. 8(a) and 8(b) shows the distances calculated by the SVR models of the further experiments. As shown in Fig. 8(a) and 8(b), the distance



(a) (b)

Fig. 7. Result of Regression model of RSSI and distance: (a) iPhone OS, (b) Android OS



(a) (b)

Fig. 8. Regression model with beacon placing on ceiling: (a) iPhone OS, (b) Android OS

predicted by the SVR model generated significant errors as the measurement environments changed.

The SVR model can not estimate the accurate distance between transmitter and receiver with these distorted RSSI values. Instead of the Regression models such as the SVR, Classification models are used in this study to locate the user more accurately.

Rather than using regression model due to anomalous Radio Frequency(RF), DNN classification models could be solution. The building was divided into many cells, and collected datasets at each cell as Fig. 2. After training the models with those datasets, cells could be classified.

In order to derive efficient localization via classification models, measurement of K-SW square building was held. For efficient localization considering the structure of K-SW, different size of rectangles were chosen for division. As shown in Fig. 2, we divided the building into many sections as follows:

- A01 - A13: normal areas including hallway
- R01 - R05: Room in K-SW building
- S01 - S09: Stair area
- E01 - E04: Exit area

Fig. 2 is the result of division of the K-SW Square building blueprint. Data was collected on mobile devices took 16 minutes per cell and stored in CSV file named with cell such as “A01.csv”. After data collection, these datas were used as input of 3 models which are Random Forest, GBM and DNN classification.

To discover the accuracy of each model, testing experiment was operated under the same conditions at different times. In TABLE V, 1.0 of accuracy with Random Forest was found, however testing accuracy was 0.6806. The maximum depth was set to 7. For GBM and DNN, the testing accuracy was plunged in comparison with validation accuracy. The hyper parameters of the GBM model is described in TABLE VI.

TABLE V
ACCURACY OF EACH MODEL

Type	Validation Accuracy	Test Accuracy	precision	recall
Random Forest	1.000	0.6806	0.6914	0.6806
GBM	0.9120	0.5595	0.7192	0.5595
DNN	0.8493	0.1874	0.1263	0.1263

TABLE VI
HYPER PARAMETERS OF GBM

Max Depth	Sub sample	Num trees	verbose
2	0.7	33	1

The problems on accuracy with testing data were found. As the user moves away from the beacon, there was a problem with the RSSI value not being as evident. The RSSI of installed all beacons were used as valid training data regardless of the distance between the beacons and the mobile device. Consequently, mobile device read the far-off beacon’s signal and the transmission cycle between the mobile phone and the beacon was mismatched, which eventually led to the data being inaccurately trained. By limiting the number of beacons to filter the valid RSSI value, this issue was fixed.

When collecting data for door localization by using DNN model, the environment is as follows.

1. If the RSSI is lower than -90, it is assumed that it is not caught.

2. Beacons are attached to the ceiling or walls more than 2m height.

3. Data is collected while person is holding device, rotating 90 degrees every 4 minutes and stores the data according to the situations. (16 minutes in total)

4. For every 30 cell, stores the RSSI of all surrounding beacons in the file named ’Tile name.csv’

Through the experiment, the accuracy of DNN was 84%. While testing under the same conditions at different times, the accuracy plunged to 18%. After considering various methods to solve this issue, it was concluded that far too many Beacons were used as the dataset. The smartphone devices read the beacon’s signal more than necessary and caught it until far-off beacons’ signals as training data. The distant Beacon was used as training data and also the transmission cycle between

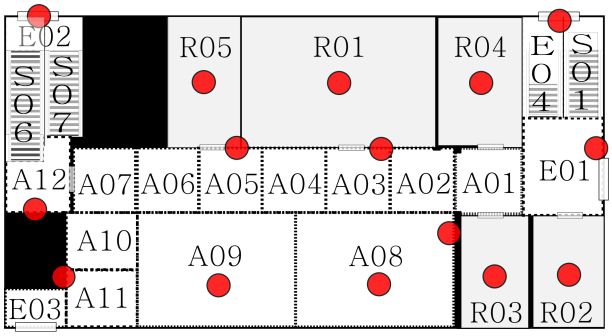


Fig. 9. Beacon placement on K-SW Square building

TABLE VII
CORRELATION BETWEEN NUMBER OF BEACON AND ACCURACY

Beacon number	loss	Accuracy	F1 Score	Highest(%)
4	0.2347	0.8853	0.6620	100
5	0.2660	0.9180	0.6935	85
7	0.2860	0.9102	0.5653	93
22	0.0238	0.9909	0.1422	14

TABLE VIII
DIFFERENCE BETWEEN KALMAN FILTERED AND RAW RSSI

Input data	loss	Accuracy	Precision	F1 Score
With Kalman Filter	0.2348	0.9148	0.9236	0.9152
Without Kalman Filter	0.6390	0.6390	0.8030	0.7357

TABLE IX
COMPARISON BETWEEN PREPROCESSED AND NON-PREPROCESSED RSSI

Preprocessed	loss	Accuracy	Precision	F1 Score
Preprocessed RSSI	0.2348	0.9148	0.9236	0.9152
Non-Preprocessed RSSI	0.4974	0.8493	0.8654	0.8478

TABLE X
ACCURACY OF MACHINE LEARNING MODEL USING 4 BEACONS

Type	loss	Accuracy	Precision	recall
Random Forest	1.000	0.7832	0.7886	0.7832
GBM	0.9067	0.7417	0.7638	0.7417
DNN	0.9148	0.7797	0.7945	0.7797

the mobile phone and the beacon was mismatched, which eventually led to the data being inaccurately learned several times.

Table VII shows the result of Accuracy, loss, F1 Score with different number of beacons used for Training.

In TABLE X, the result of the model is shown when using 4 beacons. The the accuracy of the Random forest was 1.000, which is overfitting. Between the GBM and DNN, DNN has higher precision. That is, DNN performs better when the user is expected to be in A01, but is actually there. Therefore, we were able to perform the indoor localization by using the DNN.

In Table VIII RSSI applied Kalman Filter and not applied Kalman Filter were used as input data. When using Kalman Filtered RSSI, loss was decreased by 0.2348 and accuracy was increased by 0.9148 compared to RSSI without Kalman Filter. Precision was enhanced by 0.9236 and F1 Score was improved by 0.9152. It illustrates Kalma Filter stabilized the fluctuation of raw RSSI.

We compared the cases with and without pre-processing applying the procedures described in Section III. Table IX describes DNN model accuracy for when the pre-processing is utilized and is not utilized. With pre-processing, the accuracy was improved by 0.9148. As a consequence, the indoor localization accuracy was enhanced by pre-processing method.

B. Evacuation pathfinding algorithm

The pathfinding experiment was conducted in a second-story building with a basement. The building was divided into 31 cells and 4 of exit cells. Experimental parameters for Q-learning are listed in Table XI. The number of episodes were set as 250, this could have the optimal path and also save the computational time. Based on the 250 episodes, the exploration rate shifts from 1 to 0.004. Equation 8 indicate the perfect balance between exploration and exploitation in every -0.002. The toll reward r_t was set as -0.01 to shorten the encourages the path by selecting fewer cells. The destination reward r_d was set as 1. The fire reward r_f was set as -0.8, both fire adjacent reward r_a and congested reward r_c are -0.3 by ensuring to avoid the cell. The weight fire factor F_f are set as 15 to evade the disaster. Both weight for the fire adjacent F_a and congested cell F_c are set to 3 to indicate its inefficiency. The simulation for the suggested evacuation path. When there is no fire and congestion, server provides the paths from A02 to exits and efficiency as follows.

- A02 to E01: A02 A01 E01 & 27
- A02 to E02: A02 A08 A09 A07 A12 S07 E02 & 23
- A02 to E03: A02 A08 A09 A11 E03 & 25
- A02 to E04: A02 A01 E01 E04 & 26

In a normal situation, only the length of the path was reflected to efficiency. By Equation 10. The ratio of E01's path to E04's path is 27 to 26. When the cell A01 was rewarded to r_f and the A08 was rewarded to r_c, the optimal path are shown below Fig. 10.

- A02 to E01: A02 A03 A04 A09 A07 A12 S06 S05 A13 S04 S03 S02 E01 & 11
- A02 to E02: A02 A03 A04 A09 A07 A12 S07 E02 & 19
- A02 to E03: A02 A03 A04 A09 A11 E03 & 21
- A02 to E04: A02 A03 A04 A09 A07 A12 S06 S05 A13 S04 S03 S02 E01 E04 & 10

As the logic, the ratio of E03's path to E02's path is 19 to 21. To verify the effectiveness of the proposed evacuation algorithm, Simulation was conducted in both disaster situations and normal situations. Split exits method and fastest path method was compared. We set the time for moving from one cell to the other for 1 second as well as leaving the building through the exit cell. For the disaster situation, A01 is on fire, and A08 and A09 are congested. The simulation is tested for 1/3/5/8/10

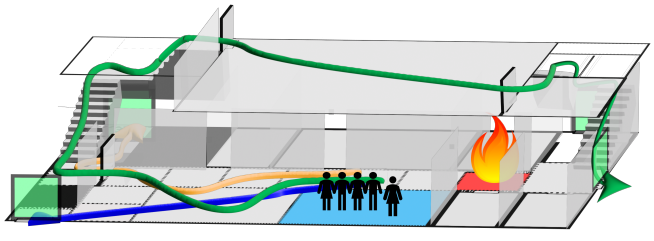


Fig. 10. Visualization of the escape path

TABLE XI
Q-LEARNING PARAMETERS

Parameter	Value
Number of episodes	250
Number of cells	31
Number of exits	4
Toll reward r_t	-0.01
Destination reward r_d	1.0
Fire reward r_f	-0.8
Fire adjacent reward r_a	-0.3
Congested reward r_c	-0.3
Fire cell scaling factor F_f	15
Fire adjacent cell scaling factor F_a	2
Congested cell scaling factor F_c	2
Pivot	30
Learning rate	0.1
Discount factor	0.8
Exploration rate	1 to 0.004

TABLE XII
EXPERIMENT RESULTS FOR EVACUATION

	Split exit	Single path
# of people in each cell	Time for evacuation (sec)	
Normal situation		
1	11.2	12.0
3	29.45	36.0
5	46.5	60.0
8	75.05	96.0
10	91.95	120.0
Disaster situation		
1	13.85	13.0
3	37.9	39.0
5	60.65	65.0
8	100.65	104.0
10	119.95	130.0

users in each cell, making a total of 31/93/155/248/310 users in the building respectively. Each simulation was tested 20 times and taken the average value. As shown in Table XII Generally split path methods are quicker than providing just optimal path method, and efficiency rises as the number of users does as well. In conclusion, split exit methods can significantly reduce the time for evacuation.

C. Guidance with Augmented Reality

We implemented navigation using 3D AR arrow and 2D map. Figure 11(b) is a screenshot of our application. Upper area displays the instruction derived from difference between heading direction of camera and destination direction. In addition, by displaying a user-friendly 3D AR arrow on the center, it was possible to guide children and foreign users who are not familiar to English. In the 2D navigation below, the current position and direction were indicated through annotation. It made user know location and direction. It also informed the evacuee the dangerous area such as congestion or fire occurred by coloring the 2D map.

V. CONCLUSION

In this paper, we have proposed Beacon-based Evacuation System & Technology using AR by using IoT devices and deep learning technologies. We designed a system that user can evacuate fast and safe without congestion. Edge devices were also adopted to check the status of each beacon, avoiding malfunctioning in emergency situation. Using the Kalman Filter and DNN model for the RSSI data allowed to estimate the indoor localization even over a large space. Based on the current user's location, the optimal evacuation route was identified in consideration of the congestion of each group of people. The Q-learning algorithm updates the reward and guides the evacuees in order to prepare for the unexpected fire situation in real-time. This paper also devised a user-friendly application by providing both 2-dimensional and Augmented Reality(AR). 2-dimensional map allows users to grasp their current location and AR navigate user step by step. We supports both Android OS and iPhone OS to support various people.

(a) (b)

Fig. 11. Screenshot of AR Navigation: (a) iPhone OS, (b) Android OS

To improve accuracy of indoor localization, support of Apple to gather RSSI of Wi-Fi should be needed. Also, to improve user performance, latest 'measure' feature on both iOS and Android OS should be needed. After user get the first location, measure feature can calculate the distance and angle from past position, that beacons in a disaster situation are no longer needed.

REFERENCES

[1] U.S. Fire Administration, "Residential building fires (2017-2019)," May 2021. [Online]. Available: Fire Estimate Summary

[2] A. Depari, A. Flammini, D. Fogli, and P. Magrino, "Indoor localization for evacuation management in emergency scenarios," in *2018 Workshop on Metrology for Industry 4.0 and IoT*, 2018, pp. 146–150.

[3] P. Kriz, F. Maly, and T. Kozel, "Improving indoor localization using bluetooth low energy beacons," *Mobile Information Systems*, vol. 2016, p. 2083094, Apr 2016. [Online]. Available: https://doi.org/10.1155/2016/2083094

[4] J. Ahn and R. Han, "An indoor augmented-reality evacuation system for the smartphone using personalized pedometry," *Human-centric Computing and Information Sciences*, vol. 2, no. 1, p. 18, Nov 2012. [Online]. Available: https://doi.org/10.1186/2192-1962-2-18

[5] Z. Zhou, T. Chen, and L. Xu, "An improved dead reckoning algorithm for indoor positioning based on inertial sensors," in *Proceedings of the 2015 International Conference on Electrical, Automation and Mechanical Engineering*. Atlantis Press, 2015/07, pp. 369–371. [Online]. Available: https://doi.org/10.2991/eame-15.2015.102

[6] A. Satan, "Bluetooth-based indoor navigation mobile system," in *2018 19th International Carpathian Control Conference (ICCC)*, 2018, pp. 332–337.

[7] Y. Zhou, Y. Pang, F. Chen, and Y. Zhang, "Three-dimensional indoor fire evacuation routing," *ISPRS International Journal of Geo-Information*, vol. 9, no. 10, 2020. [Online]. Available: https://www.mdpi.com/2220-9964/9/10/558

[8] F. Mirahadi and B. Y. McCabe, "Evacusafe: A real-time model for building evacuation based on dijkstra's algorithm," *Journal of Building Engineering*, vol. 34, p. 101687, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352710219324982

[9] S. Sadowski and P. Spachos, "Optimization of ble beacon density for rssi-based indoor localization," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2019, pp. 1–6.

[10] C. H. Lam and J. She, "Distance estimation on moving object using ble beacon," in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2019, pp. 1–6.

[11] AltBeacon, "Altbeacon/android-beacon-library: Allows android apps to interact with ble beacons." [Online]. Available: https://github.com/AltBeacon/android-beacon-library

[12] K. Shimizu and D. Kushida, "Evacuation guidance system using beacon information and dijkstra's algorithm," in *2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*, 2021, pp. 319–323.

[13] A. Khan, A. A. Aesha, J. S. Aka, S. M. F. Rahman, and M. J.-U. Rahman, "An iot based intelligent fire evacuation system," in *2018 21st International Conference of Computer and Information Technology (ICCIT)*, 2018, pp. 1–6.

[14] U. Atila, Y. Ortakci, K. Ozacar, E. Demiral, and I. R. Karas, "Smartescape: A mobile smart individual fire evacuation system based on 3d spatial model," *ISPRS International Journal of Geo-Information*, vol. 7, no. 6, p. 223, 2018.

[15] M. Bhargava, *Beacons With Raspberry Pi. In IOT projects with Bluetooth Low Energy: Harness the power of connected things*. Packt, 2017.

[16] M. Lipka, E. Sippel, and M. Vossiek, "An extended kalman filter for direct, real-time, phase-based high precision indoor localization," *IEEE Access*, vol. 7, pp. 25 288–25 297, 2019.

[17] U. Atila, Y. Ortakci, K. Ozacar, E. Demiral, and I. R. Karas, "Smartescape: A mobile smart individual fire evacuation system based on 3d spatial model," *ISPRS International Journal of Geo-Information*, vol. 7, no. 6, 2018. [Online]. Available: https://www.mdpi.com/2220-9964/7/6/223

[18] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," in *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, 2015, pp. 74–77.

[19] J. Gu, M. Zhu, Z. Zhou, F. Zhang, Z. Lin, Q. Zhang, and M. Breternitz, "Implementation and evaluation of deep neural networks (dnn) on mainstream heterogeneous systems," in *Proceedings of 5th Asia-Pacific Workshop on Syste ms*, 2014, pp. 1–7.

[20] J. He, L. Li, J. Xu, and C. Zheng, "Relu deep neural networks and linear finite elements," *arXiv preprint arXiv:1807.03973*, 2018.

[21] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *Advances in neural information processing systems*, vol. 29, 2016.