Foodflare: An Indoor Navigation System

M. Varalatchoumy(✉), Santosh Divakaran, and R. Abhishek Ram

Cambridge Institute of Technology, Bengaluru, India

Abstract. GPS-based navigation has been the go to approach for outdoor navigation systems/applications whereas is deemed ineffective and unreliable in indoor environments. An ideal indoor navigation system requires extensive hardware installations and high setup costs. In this paper, we present FoodFlare a cost effective indoor navigation guidance system for retail environment implementation that minimises the need of human assistance and the need of existing static maps. The proposed system uses the existing sensors built in a mobile device like camera, gyroscope, accelerometer, magnetometer and integrates with Augmented Reality to provide the user with a fluid experience.

Keywords: Augmented reality · Indoor navigation · Indoor positioning

1 Introduction

Augmented reality is a combination of the real world and a computer-generated virtual world. This is achieved by enriching computer-generated images with the real world. [1] Presently, AR is available on most mobile devices. Foodflare is an application developed to reduce everyday hassle by giving turn by turn instructions for users to navigate in indoor spaces. Foodflare is independent of any external hardware, it is fully functional with already available hardware. The use of GPS indoors is unreliable due to a variety of reasons.Foodflare excels in locations where GPS fails, such as within buildings where a GPS signal is unreliable variety of reasons including poor signal strength, multipath effect and limited on-device computation and communication power [2]. Foodflare enables dynamic indoor localisation, providing high accuracy and quick routing, offering the shortest minimal optimal path. Navigation is done by accepting a destination from the user. The application then localises the user's location, and provides navigational cues for simplicity.

2 Literature Survey

The primary requirement for using indoor navigation involves building a map of the location. There are an increasing number of mapping algorithms that are optimised for indoor navigation, such as [18] which uses a two level routing approach, but this is an algorithm that works on pre existing floor maps. Meghna Pal [3] et al. implements a method which uses GPS and maps data, using which locations are marked, this approach is simple to implement, but faces issues with accuracy, and inability to function indoors.

S. Tamane et al. (Eds.): ICAMIDA 2022, ACSR 105, pp. 722–734, 2023.
https://doi.org/10.2991/978-94-6463-136-4_62

A low power system built in [19] uses the phone's inbuilt compass and accelerometer to estimate the users location in an indoor space based on their heading and steps taken. Although this approach can introduce a lot of variations due to gait length, systems such as [20, 21] can be adopted to get the actual step length. Binu P K [4] et al. overcome those issues by building a map by collecting magnetic field data from the phone's magnetic sensor, additionally the map data is generated automatically based on the paths taken by the devices, but this system requires additional hardware setup. A similar approach is implemented by [14], which also incorporates human motion analysis for reduced error. Pascal Bissig [5, 12] gathers information from WLAN signal strength of nearby routers and generates a map by consolidating the information. Similarly, [13] gathers information from RSSI fingerprints and google maps data. This system is more viable as more locations have pre-existing wireless networks, but it requires multiple routers for a reliable connection. The crowdsourcing system implemented by [11, 25] helps reduce the requirement of additional hardware by using the crowd information to help with location marking. Another approach to crowd sourced data is shown in [22] where the system builds a floor map using recorded video data from various participants. But crowd sourced data also inherits a greater security risk, as shown in [23]. Tanishq Gupta and Holden Li [6] implement an approach using the Xbox Kinect and ZED camera. The mean error from the ZED camera was noted to be 13.87% while the Kinect controller had an error rate of 3.7%. It was also noted in [15] that the RMS error is about 48.8 mm from a distance of 3 m. The implemented system requires additional proprietary hardware on the user end, and is not easily installable. Ibrahim Alper Koc [7] et al. used Augmented reality for mapping and navigation, which had an average error of 0.29 m indoors and 7.8 m outdoors.

Localisation is needed for determining the position of the user in the given map; this has been done utilising various approaches. One approach taken is using the radio frequencies emitted by a device, and localising a location using it as a synthetic aperture radar, this however has not been implemented in a mobile operating system. A simpler approach was implemented in [17] where a static map of the location is built, and various markers are placed throughout the location. The user's location is updated based on the marker scanned by them. Bo-Chen Huang [8] has implemented an indoor navigation system using Bluetooth low energy beacons, assisted by AR. This system uses the beacons for primary positioning, while providing visual guidance by the means of AR overlays on the camera feed. Gaurav Gupta [9] demonstrates an indoor navigation system which utilises a smartphone and a backend server. In this system, the backend server does the majority of the processing.. Ibrahim Arda Cankaya [10] took user input for the starting location and then used AR for navigating them to their desired destination. Calculation of the user's position in the path was done by combining filtered accelerometer data, user gender, and user height.

3 System Overview

3.1 Architectural Design

The AR indoor navigation application is developed using Unity and Xcode with integration of the Placenote SDK. The Placenote SDK has a cloud service functionality which

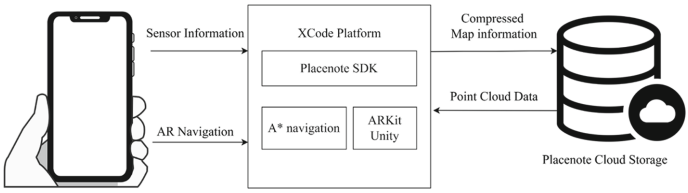

Fig. 1. The figure shows the architecture of Foodflare. The data collected by the AR Scanning tools are converted to a point cloud and stored in the cloud storage. This data is then retrieved for navigation locally.

for our application is used to store AR maps. The AR Point cloud data is collected by the application during the mapping phase, camera input, gyroscope, accelerometer and geo magnetic sensor data are used for generation of the point cloud. The point cloud data, along with other manually added points of interest are stored. Upon completion of mapping, the map is saved to the Placenote cloud storage. When the application is then used for navigation, the map data is fetched from the Placenote cloud storage, and the data is used for localisation and navigation. All processing is done locally on the device, and the cloud storage is used only for storing maps. This architecture helps keep latency to a minimum, while also not requiring any significant server infrastructure (Fig. 1).

3.2 System Operation

Upon launching the application, the Placenote SDK is first initialised. Upon successful initialisation of the SDK, the user can either load a pre existing map from the placenote database, or create a new map.

For the creation of a new map, the application starts gathering spatial data from the sensors. In addition to gathering raw spatial data, the application also allows the user to add pointers and destinations. Pointers are manually added and are used as the nodes on which the pathfinding algorithms find the shortest path from a source to a destination. Pointers must be added intermittently, and should be added in crosspaths, junctions or any location where a change in direction is expected. This ensures more fluid navigation during the navigation phase.

For loading of an existing map, the user is initially prompted with a dropdown menu to choose the appropriate map to be downloaded, the application then starts gathering spatial information from the sensors, after which the user can choose to either extend an existing map or to load and start navigation, a dropdown menu is prompted to select the desired destination location to be navigated. The user at any point can choose to view all the destination nodes in the current loaded map (Figs. 2, 3 and 4).

3.3 User Interface

The applications user interface enables the user to easily navigate through the application. The buttons in the interface are kept to a minimum, as the remaining space acts as a viewfinder for the user.

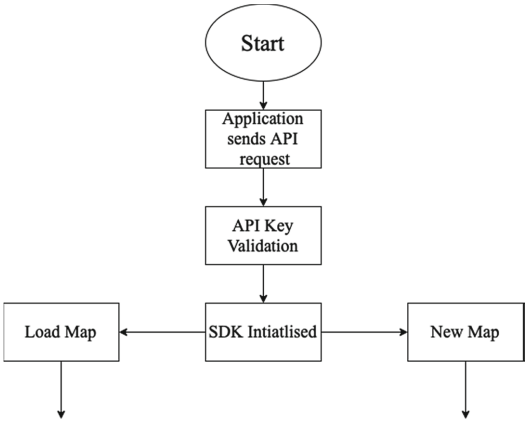

Fig. 2. Shows the SDK initialisation process and the options presented to the user upon successful initialisation.

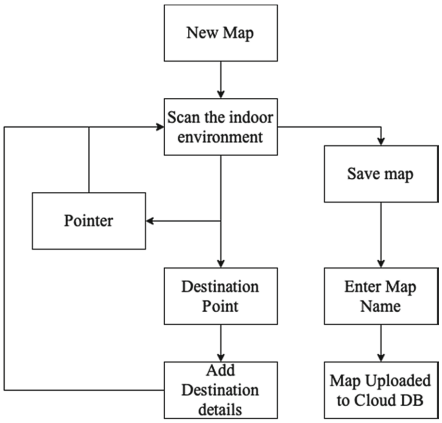

Fig. 3. Illustrates the process of creating a new map, adding destinations and pointers in a new map

When the user is creating a map, the point cloud is also represented by using coloured dots. This helps the user understand if a specific region has been mapped, and also acts as an indicator of the level of detail present (Figs. 5, 6 and 7).

When the user wants to load an existing map, they select it from a list available to them, after which a popup is presented, indicating the application is gathering data to try and localise the user. Once localised, the user is given a list of destinations to select from.

Once the user selects a destination, the A* algorithm determines the shortest path to the destination, and the path is calculated based on the pointers added during the mapping phase. While navigating, the pointers are overlaid on the viewfinder, in the exact location they were created. The pointers are rendered as an arrow which points to

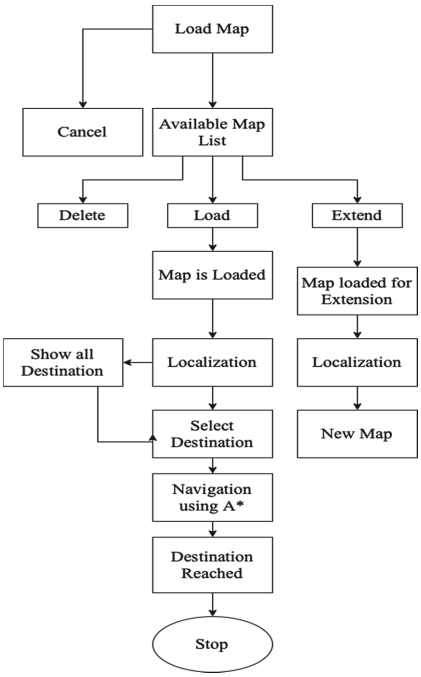

Fig. 4. Illustrates the process of loading an existing map, gathering information to localise the users position in the map, and provide the navigational information to the destination, using the shortest path.

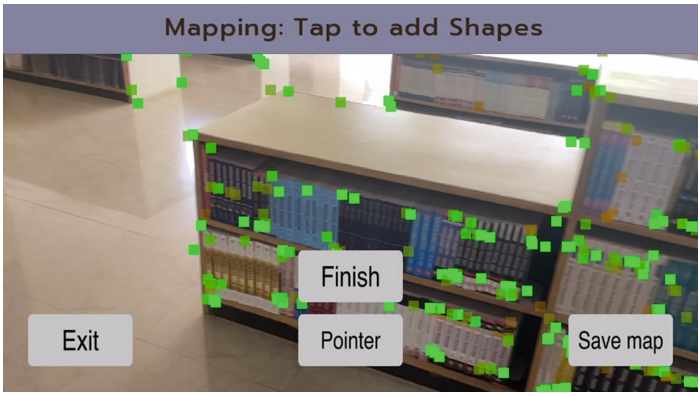

Fig. 5. Shows the interface presented to the user during mapping

the next pointer in the navigational path. Not more than 2 pointers are rendered at any single point of time, this helps both conserve resources as well as mitigate any issues with occlusion.

Fig. 6. Shows the process of the application trying to localise the users position within the map

Fig. 7. Shows the process of navigating the user to the destination, including the pointer to the destination, and the destination beacon itself.

4 Methodology

4.1 System Specifications

The application requires Placenote SDk supported for iOS 12 and above devices.

- The device requires a minimum RAM of 734 MB
- A camera is needed for the AR and nearby discovery features to function.
- Gyroscope and accelerometer information is also needed for ARKit

4.2 Augmented Reality

Augmented Reality (AR) is an augmented version of the real physical world, achieved using digital visuals, sounds, or other sensory stimuli provided by technology. AR is a view of an actual, real-world environment, overlaying computer-generated imagery to change our understanding of reality.

The system we propose uses augmented reality to display arrows and directions that users can see on their phones. The following software development kits are used:

- Unity
- Placenote SDK

Unity is the tool many developers use to build and power their creations. It's a 3D/2D game engine and a powerful cross-platform IDE for developers. Take 2D, 3D, art and assets, assemble these materials into scenes and environments, add sound, lighting and other special effects, physics and animation, interactivity and game logic, and tailor them to your capabilities and targets. to edit, monitor, and optimise content. Platform. Unity3D provides powerful tools for creating immersive and augmented reality experiences that intelligently interact with the real world.

Placenote SDK lets people build easy cloud deployable Augmented Reality (AR) app, It leverages visual mapping in a way to save and share persistent AR content in physical spaces Placenote does not need GPS, markers or beacons for geolocation. Instead, it lets you scan any space and turn it into a smart canvas for positioning AR objects. Placenote can easily be integrated into Unity or native iOS apps, and development can be either done using Unity 3D or Swift/Native iOS platforms.

A* Algorithm was developed based on the Dijkstra algorithm. Starting from a particular node, the current child node's weight value is updated, and the child node with the lowest weight value is used to update the current node until all nodes have been traversed. The key to the A* algorithm is to set the score function f(n), $f(n) = g(n) + h(n)$. Where g(n) represents the actual cost from the starting node to node n. h(n) represents the estimated cost of the best path from node n to the destination node in the state space. The Euclidean distance between two nodes is usually taken as the value of h(n). For a constant value of g(n), the value of f(n) is primarily affected by the value of h(n). If the node is close to the target node, the value of h(n) will be small and the value of f(n) will be relatively small. This ensures that the shortest route search is always performed in the direction of the target point. The A* algorithm searches along the target points, taking into account the location information of the target points of the mobile robot. The path search efficiency of the A* algorithm is higher compared to the Dijkstra algorithm.

The A* algorithm implemented in Foodflare uses a Manhattan distance heuristic. The Manhattan heuristic is based on a type of distance calculation called the Manhattan distance. The Manhattan distance, also known as city block or taxi cab distance is a distance metric between two points. It's the sum of the absolute differences between these points' coordinates.

The navigation in Foodflare is independent of the order of nodes placed. As shown in Fig. 8, despite the location being mapped starting from location A and progressing sequentially up to location L. Despite this, if a user localises the application from a

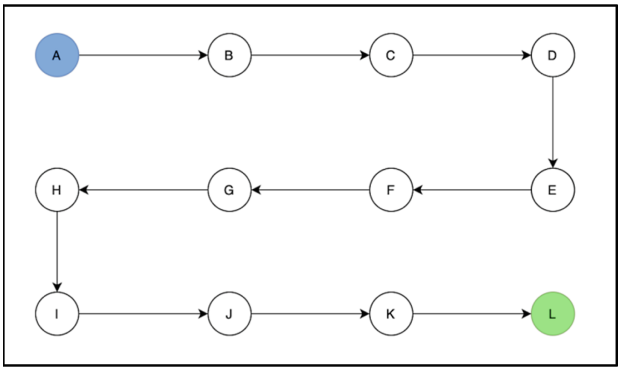

Fig. 8. Shows a sample sequence of adding destinations in a map where points A to L are destinations.

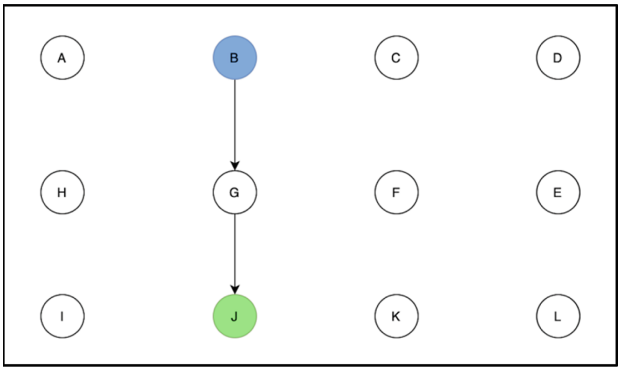

Fig. 9. Shows a case of a user starting navigation from a random location, and the application navigating to the destination using the shortest path.

different area such as B, and navigates to position J, the algorithm navigates using the shortest path which is from B → G → J (Fig. 9).

5 Experimental Results

The performance of the application is a key factor that must be taken into consideration as a resource intensive application is less likely to run on a lot of low power devices, ultimately making it unfavourable to the user. To gauge the performance of the application, the following three key metrics are tracked:

- Accuracy of the navigation
- Memory consumption
- Battery consumption

Table 1. Accuracy of Foodflare

Distance (m)	Lighting conditions	Accuracy (m)
50	Well lit	0.1–2.5
50	Dimly lit	0.17–0.4
200	Well lit	0.16–1.2
200	Dimly lit	0.37–2.1

5.1 Accuracy

The accuracy of the navigation offered by foodflare varies based on the environment it is tested in, along with the lighting conditions of the environment. Through a variety of tests, it was concluded that the application has the highest accuracy in relatively smaller, well lit areas. The application had an accuracy ranging from 0.1 to 0.25 m while navigating in a well lit environment, where the navigation distance was approximately 50 m. For the same environment, when tested in lower lighting conditions, a slight drop in accuracy was observed, ranging from 0.17–0.4 m.

When tested for a longer navigation distance of 200 m, there was a slight drop in accuracy. In a well lit environment, the highest accuracy had an error of 0.16 m, but ranging upto 1.2 m. The drift was more evident for long distance navigation in a dimly lit environment, where the accuracy was ranging from 0.37 m, all the way up to 2.1 m. This occurs mainly due to the primary dependance of AR on the devices front camera.

In environments with lower illumination, it is harder to identify features, and hence, less usable information is present for navigation. The drift in navigation, however, is greatly reduced when a point is re-localised near the destination. This allows the application to recalibrate itself to its surroundings. The Observed data is shown in Table 1.

5.2 Memory Consumption

Computer memory is the storage space within a computer where data needs to be processed and the instructions needed for processing are stored. It is a crucial and limited resource and the overconsumption of it can lead to the device crashing, or having diminished performance. High memory utilisation can also cause higher power consumption, which is not ideal on a mobile device, as that leads to overheating and quick depletion of the battery.

The memory consumed by Foodflare is dependent on the tasks performed by it, since some tasks have a higher memory requirement than others. While creating a map, the memory requirement is higher than that of navigation. This is due to the application storing the point cloud data on the device until the mapping is complete.

The memory consumption also varies based on the duration of the scanning, as a longer scan inherently results in a higher level of point cloud data being generated. During scanning for a short period of time, the memory consumed ranged between 380–470 MB. In similar conditions, while scanning an area for a prolonged period of time, the memory consumption varied between 500–700 MB.

Table 2. Memory consumed by Foodflare

Action	Duration of working	Memory consumption (MB)
Scanning	5 min	380–470
Scanning	20 min	500–700
Navigating	5 min	320–410
Navigating	20 min	390–510

Table 3. Battery consumed by Foodflare

Task	Duration (mins)	Battery consumed
Scanning	90	27%
Navigation	90	25%
Video recording	90	20%
Scanning	90	27%

The process of navigating consumes less memory in general, as it only needs to keep a portion of the relevant map in memory. In addition, only a limited number of nodes were rendered at a singular point in time. In a short navigation period, the memory consumed ranged between 320–410 MB, and for a longer duration navigation, the memory consumption ranged between 390–510 MB. The observed data is shown in Table 2.

5.3 Power Consumption

All mobile devices have a limited amount of battery available, and this resource must be utilised diligently. If a single application consumes a significant amount of power, it results in the device not being able to operate for a prolonged period of time. AR applications have typically consumed high amounts of power. But the advancements in processor technology and software optimisations have made AR applications significantly more energy efficient.

Scanning and navigation both consumed very similar amounts of power during a 90 min window, where they drained 27% and 25% power respectively. For reference, recording a video for 90 min resulted in a power consumption of 20%. Although both tasks require continuous access to the camera, AR applications require a higher amount of processing to interpret the data, this results in a greater power consumption. Since the expected application use time per session is not expected to be more than 20 min, the impact on the battery is not very significant. The battery consumption data is shown in Table 3.

6 Conclusion and Future Work

In summary, FoodFlare is an indoor navigation guidance system that uses mobile's built-in sensors and AR technology to provide users with navigation cues and provide a smooth experience. It successfully detects the user's current location and displays AR navigation instructions on top of the actual camera view to help the user navigate to their destination.

The current prototype is designed for same-floor navigation. Future work can be done to extend to multi-level navigation. Multi-story mapping and positioning are already implemented in the application, so multi-floor navigation can be easily achieved by modifying the AR cues.

In the future, we also plan to introduce a feature that allows existing users to rate the routes they have taken within the internal structure, allowing other users to select the most rated route to their destination. Doing. A voice support feature can also be implemented for users who prefer it.

References

1. R. Aggarwal and A. Singhal, "Augmented Reality and its effect on our life," 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2019, pp. 510–515, doi: https://doi.org/10.1109/CONFLUENCE.2019.8776989.
2. S. Nirjon, J. Liu, G. DeJean, B. Priyantha, Y. Jin, and T. Hart, "Coin- GPS: indoor localization from direct GPS receiving," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pp. 301–314, ACM, 2014.
3. M. Pal, A. Thakral, R. Chawla and S. Kumar, "Indoor maps: Simulation in a virtual environment," 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), 2017, pp. 967-972, doi: https://doi.org/10.1109/SmartTechCon.2017.8358515.
4. P. K. Binu, R. A. Krishnan and A. P. Kumar, "An efficient indoor location tracking and navigation system using simple magnetic map matching," 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2016, pp. 1-7, doi: https://doi.org/10.1109/ICCIC.2016.7919537.
5. P. Bissig, R. Wattenhofer and S. Welten, "A pocket guide to indoor mapping," 2013 10th Workshop on Positioning, Navigation and Communication (WPNC), 2013, pp. 1-6, doi: https://doi.org/10.1109/WPNC.2013.6533272.
6. T. Gupta and H. Li, "Indoor mapping for smart cities — An affordable approach: Using Kinect Sensor and ZED stereo camera," 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2017, pp. 1-8, doi: https://doi.org/10.1109/IPIN.2017.8115909.
7. A. Koc, T. Serif, S. Gören and G. Ghinea, "Indoor Mapping and Positioning using Augmented Reality," 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), 2019, pp. 335–342, doi: https://doi.org/10.1109/FiCloud.2019.00056.
8. B.-C. Huang, J. Hsu, E. T.-H. Chu, and H.-M. Wu, "ARBIN: Augmented Reality Based Indoor Navigation System," *Sensors*, vol. 20, no. 20, p. 5890, Oct. 2020, doi: https://doi.org/10.3390/s20205890.
9. G. Gupta, N. Kejriwal, P. Pallav, E. Hassan, S. Kumar and R. Hebbalaguppe, "Indoor Localisation and Navigation on Augmented Reality Devices," 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct), 2016, pp. 107–112, doi: https://doi.org/10.1109/ISMAR-Adjunct.2016.0052.

10. A. Cankaya, A. Koyun, T. Yigit and A. S. Yuksel, "Mobile indoor navigation system in iOS platform using augmented reality," 2015 9th International Conference on Application of Information and Communication Technologies (AICT), 2015, pp. 281–284, doi: https://doi.org/10.1109/ICAICT.2015.7338563.
11. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: zero-effort crowdsourcing for indoor localization," ser. Mobicom. ACM,2012, pp. 293–304.
12. Dongsoo Han, A Sensor Fusion Method For Wifi Based Indoor Positioning, ICT Express 2 (2016) 71–74
13. Sagar V. Ramani, Yagnik N. Tank, Indoor Navigation on Google Maps and Indoor Localization Using RSSFingerprinting, International Journal of Engineering Trendsand Technology (IJETT) – Volume 11 Number 4 - May 2014
14. Gerald Glanzer, Self-Contained Indoor Pedestrian Navigation By Means Of Human Motion Analysis and Magnetic Field Mapping, 2010
15. K. K. a. S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping," Sensors, 2012.
16. Abhijit Chandgadkar, "An Indoor Navigation System For Smartphones", 2013.
17. Mulloni, A.; Wagner, D.; Barakonyi, I.; Schmalstieg, D. Indoor positioning and navigation with camera phones. IEEE Pervasive Comput. 2009, 8, 22–31.
18. L. Liu, S. Zlatanova, " A "door-to-door" Path-finding Approach for Indoor Navigation," Proceedings Gi4DM: GeoInformation for Disaster Management, May 3–8, 2011.
19. J. B. Link, P. Smith, N. Viol and K. Wehrle, "Footpath: Accurate map-based indoor navigation using smartphones,"In Indoor Positioning and Indoor Navigation (IPIN),International Conference IEEE, pp. 1–8, September 21–23,2011.
20. L. Klingbeil, et al., "A modular and mobile system for indoor localization," Intl. Conf. Indoor Positioning & Navigation,2010, pp. 1–10.
21. P. Davidson, J. Collin, J. Takala, "Application of particle filters for indoor positioning using floor plans", Proceedings of Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS),pp. 1–4, Kirkkonummi, Finland, October 2010.
22. S. Chen, M. Li, K. Ren, and C. Qiao, "CrowdMap: Accurate recon-struction of indoor floor plans from crowdsourced sensor-rich videos,"in IEEE ICDCS, Columbus, OH, June 2015.
23. T. Li, Y. Chen, R. Zhang, Y. Zhang, and T. Hedgpeth, "Secure crowd-sourced indoor positioning systems," in IEEE INFOCOM, Honolulu, HI,Apr. 2018.
24. S. Kumar, S. Gil, D. Katabi, and D. Rus, "Accurate indoor localization with zero start-up cost," in ACM MobiCom, Maui, HI, Sep. 2014.
25. Chen, Yuanfang & Shu, Lei & Ortiz, Antonio M. & Crespi, Noel & Lv, Lin. (2014). Locating in Crowdsourcing-Based DataSpace: Wireless Indoor Localization without Special Devices. Mobile Networks and Applications. 19. 1-9. https://doi.org/10.1007/s11036-014-0517-8.