Towards An Indoor Navigation System Using Monocular Visual SLAM

Akshat Bajpai
Dept. of Computer Science
University of the Pacific
Stockton, CA, USA
a_bajpai@u.pacific.edu

Sepehr Amir-Mohammadian
Dept. of Computer Science
University of the Pacific
Stockton, CA, USA
samirmohammadian@pacific.edu

***Abstract*—This paper presents a novel implementation of an augmented reality-aided indoor navigation system for mobile devices. The proposed system uses the device's camera to scan the environment, generate an abstracted 3D map of the indoor environment, and transfer it to a remote server. The 3D map of the indoor environment is achieved through a tracking and mapping ARKit module. Once the indoor map is stored in the server, it can be accessed simultaneously by multiple devices for localization and navigation. Leveraging Unity assets and the directions retrieved from the server, the application computes the shortest distance between the source and destination, and displays AR-based direction markers for navigation assistance.**

***Index Terms*—Augmented reality, Indoor navigation systems, Simultaneous localization and mapping**

I. Introduction

In recent years, the problem of navigating an object has been a popular topic of research. While outdoor navigation has reached a commercial-level efficiency [1], indoor navigation systems (INS) are still trailing the success of their outdoor counterparts. Outdoor navigation systems have gained popularity using inertial-GPS trackers since the early 2000s [2]. The most widely used commercially available indoor navigation systems use Bluetooth Lite (BLE) beacon technology [3]. However, BLE beacons suffer in terms of efficiency compared to the techniques that are common in outdoor navigation systems. Wi-Fi, lidar, and infrared sensors are also being used for indoor navigation [4].

Among different approaches to implement INSs, vision-based solutions are appealing as they provide better system usability and operability. Vision-based techniques resonate well with how humans identify physical environments, i.e., by associating various unique landmarks and objects within the environment. A vision-based INS usually includes a network of unique markers [5]. These markers could be QR codes, barcodes, ArUco markers or other customized patterns [6]. The user navigates their way through the environment, scanning the identification markers with their mobile cameras. While this approach provides a straightforward implementation with high accuracy, it is highly dependent on user's accessibility to these markers. If a marker is not placed in an easily-approachable location, finding the subsequent markers could be time-consuming and/or confusing. Furthermore, establishing a new vision-based INS in an unfamiliar location can be quite tedious. A better alternative is to use simultaneous localization and mapping.

Simultaneous localization and mapping (SLAM) is a computational problem that tries to realize whether it is possible for a mobile machine to be placed in an unknown environment and incrementally construct a 3D representation of that environment while simultaneously localizing itself in that environment [7]. The applications of SLAM include indoor cleaners, self-driving vehicles, autonomous robots, and in particular extended reality [8]. With the advancements in the smart mobile device camera technologies, both iOS and Android operating systems have incorporated native augmented reality [9] software development kits, viz. ARKit [10] and ARCore [11], respectively. However, the accuracy of these systems and the complexity of conditions for creating maps and installing technologies that could be leveraged for navigation purposes has been challenging [12], e.g., wrt storage and retrieval of 3D point clouds.

There are several methods of implementing SLAM such as extended Kalman filter [13], parallel tracking and mapping [14], large-scale direct monocular SLAM [15], and Oriented-FAST and Rotated-BRIEF SLAM [16]. One of the key features for a solution to the SLAM problem is estimating the unknown values using known ones. The known values are sensor-based values taken from the environment. In the case of visual SLAM, various types of sensors could be used, e.g., RGB cameras, lidar, radar, monocular cameras, and stereo cameras. The input stream of values from these sensors are then fed to a feature extraction algorithm e.g., Scale Invariant Feature Transform [17], Speeded Up Robust Features [18], and Oriented-FAST and rotated-BRIEF [19]. The most commonly used features are corners, intersections, edges, and color intensity. These extracted features are then passed on to the pose-estimation phase, where the movement of these features is tracked in the input stream. Through the movement of the extracted features, a 3D map is created. Finally, loop closure and bundle adjustment steps are used to refine the constructed map. Figure 1 demonstrates the steps through which visual SLAM process constructs and refines the 3D map.

In this paper, we propose a markerless, vision-based, cost-effective, and realtime solution for indoor navigation using visual SLAM. The ability of visual SLAM to work on monocular

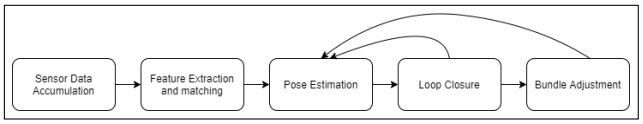

Fig. 1. General steps in visual SLAM.

camera systems is essential to the proposed system, since our proposed system is aimed to work on a mobile device's built-in camera. Advancements over the years in computer vision algorithms have shown the high potential of visual SLAM in a mobile device that possesses a monocular camera [20]. For the sake of scalability, and considering the aforementioned challenges associated with AR SDKs on mobile platforms, we have employed a recently incorporated ARKit module, known as ARWorldMap [21]. ARWorldMap allows users to create location-based 3D point cloud maps, by constructing objects that contain the snapshot of all relevant information, e.g., coordinates and object identifiers.

Contributions. INSs are still maturing [22]. In this paper, we are aiming to overcome the limitations of the existing methodologies using vision-based techniques with the features given in the following. Ultimately these features aim to boost the usability and operability of our system in comparison to other proposed INSs.

- Markerlessness: One of the common and prominent limitations of the existing vision-based systems is the constant need for scanning the markers after every few steps. This results in a relatively slow traversal process. Our proposed system is an implementation of vision-based INS that completely eliminates the need for markers, hence being markerless.
- Efficient initial positioning: Yet another limitation of existing vision-based systems is localizing the initial position of the device. For instance, in a marker based INS, if the device is not placed in the right place, finding the location of the nearest marker can be a challenging task. This is a significant usability concern. Leveraging a SLAM-based solution tackles this issue.
- Realtimeness: As the client launches the application on their mobile device, the device quickly scans the environment on the go, and is able to perceive its location by matching the current image frames with the pre-stored 3D map of the environment on the server. This entails realtime localization.
- Cost-efficiency: With the advancements in the computational capabilities and built-in cameras of smart mobile devices, the need to acquire additional hardware for scanning purposes is eliminated. Modern mobile devices process many of the computationally-intensive operations that used to be offloaded to remote services due to lack of resources.
- Reducing traversal speed: Being markerless provides a more seamless and faster navigation experience for the end user compared to the marker-based alternatives. This eliminates the need to be in the vicinity of the marker's location for scanning it. Additionally, the client-server pipeline is only used to communicate the key points. The rest of the computation takes place within the mobile device. This entails a significantly faster solution compared to the ones dependent on server-side computations.
- Security and privacy: The client-server pipeline is secured with TLS protocol for exchanging localization coordinates. Moreover, the exact location of the client and the destination point are not revealed to the server, as the server blindly sends the whole 3D map of the current environment to the client.

Paper outline. In Section II, we review different technologies that have been used in the wild to deploy INSs. In Section III, we specify our proposed system in a high-level manner, and discuss our prototype implementation. Section IV reviews the related work, and finally Section V concludes the paper and specifies future work.

II. Indoor navigation systems

In this section, we first distinguish INSs from Indoor Positioning Systems (IPSs). Next, we provide a brief survey of existing technologies in INSs.

A. Indoor positing and navigation

Positioning is a process by which a system understands its location wrt an identified reference frame. This reference frame usually consists of a network of pre-calibrated stationary devices or markers. For outdoor environments, GPS satellites are used for accurate localization. However, using GPS in indoor environments is challenging due to signal attenuation and reflection. Wi-Fi/Li-Fi, BLE, and infrared transmitters for indoor positioning deal with the same challenges, as well. These issues are eliminated when it comes to vision-based positioning.

An indoor navigation system may use an indoor positioning system to identify the initial location of a device. Navigation computes a path from the initial location of the device to the desired destination, and helps the device using sequential step-by-step directions to traverse that path. The path is usually calculated according to the distance, traffic, number of obstacles along the path, and other environment-specific custom parameters.

B. Existing technologies in INSs

The existing technologies for implementing an INS are broadly classified into two categories: internal sensory based systems, and external environment based systems.

1) Internal sensory based INSs: Nowadays, the majority of smart mobile devices are equipped with several sensors, collecting ever-evolving environmental data. These sensors take quantitative readings of information including the inertial, acceleration and magnetic forces. The accelerometers are used to measure the inertial changes to sense motion. Interdevice sensors such as gyroscopes and magnetometers measure the change in the magnetic fields to sense values such as orientation, velocity, and position of the device. The

most widely adapted approach to work with interdevice sensor data uses the dead-reckoning algorithm [23]. The advantages of using these built-in sensors include negligible extra cost of implementing IPSs. Employing these sensors provides some level of performance efficiency even after there is a change in the external environment. This entails environmental independence. In addition, it provides some level of privacy, since all computations are performed locally on the device. On the other hand, accuracy of these sensors is questionable. Localization only works after determining the initial position, and any change that affects the magnetic fields of the environment can result in incorrect interpretation of location [4].

2) External environment based INSs: The external environment based technologies highly depend on the environmental cues, e.g., signal strength, marker-based location tags, etc. A network of these cues is constructed for a specific area. In what follows, we describe two types of external environment based systems considering the environmental cues that are used.

a) Signal information based systems: These systems are comprised of signal transmitters and receivers, where one part is kept stationary (usually transmitters) and the other component is dynamic. The dynamic component collects signal information such as strength, time of arrival, time-difference of arrival, angle of arrival, etc. Based on these values, algorithms such as trilateration [24], and Bancroft [25] are used for positioning. Wi-Fi [26], Li-Fi [27], BLE beacons [3], near-field communication [28], and infrared sensors [29] fall under this category. These systems provide increased accuracy. If the signal emitting devices are static and calibrated, localization is independent of any other change in the environment. Moreover, due to a client-server like protocol, realtime monitoring of localization is possible, which may introduce privacy concerns. These systems are prone to error due to signal attenuation and reflection in indoor environments. In addition, device calibration is not trivial and it has to stay static once configured. Another disadvantage is that the entire network of transmitters is costly.

b) Vision-based systems: This approach is gaining more popularity due to the recent advancements in AR technology. The device built-in cameras are used to recognize environmental marker-based and markerless cues. Most commonly, QR codes, barcodes, or custom markers are employed to recognize the indoor locations [4]. This approach is comparatively cheaper than the rest of the techniques for INSs, as it does not require additional hardware. It is also highly accurate and maintains user privacy, e.g., regarding user's location. However, the marker-based systems require the users to constantly move close to the markers for scanning as they traverse the path, which is challenging for timely navigation. Moreover, the camera resolution is influential in the quality of the service.

III. Proposed System

Our proposed system [30] consists of two phases: 1) Generating 3D map of the environment, and 2) localization of the device using the ARWorldMap coordinates. Figure 2 demonstrates the high-level architecture of the system which consists of different functionalities being invoked by the smart mobile devices (clients) and different components of the server to accomplish navigation.

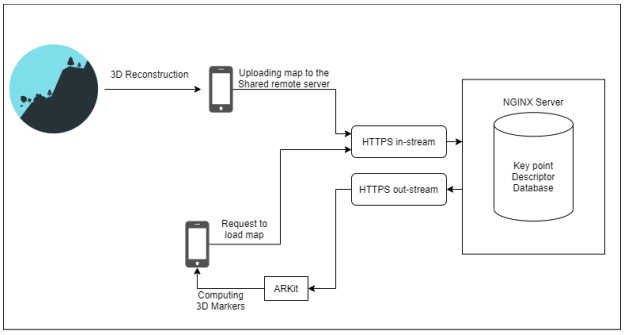

Fig. 2. The proposed system architecture.

A. Generating a 3D point cloud

Initially, a 3D map of the environment is constructed for the purpose of localization in the next phase. The client-side application scans the environment and places the location tags on the generated structure according to the user requests. The shortest route between the location tags is computed simultaneously using A* algorithm. This is done by developing a customized, feature-based keypoint point cloud in an ARWorldMap format. This structure, called keypoint descriptor, is then stored in a shared remote server. The format of the descriptor is as follows.

```
{ Key_point ID:
  [ARWorldMap_file, location_tags] }
```

Each element in this database has a `Key_point ID` which is an arbitrarily generated unique identifier of the environment. The values associated with an identifier include the 3D map of the environment that is described by a collection of keypoints (in `ARWorldMap_file`) and a list of `location_tags`. Location tags consist of all the locations of interest, identified by the user at the time of generating the 3D map. These tags are potential source and destination points for navigation. The shortest path among the location tags is stored in `ARWorldMap_file` using a spanning tree structure.

B. Localization and navigation

Once the environment is scanned and the keypoint descriptor is uploaded to the server, the system is ready to step into the second phase, i.e., localization of different smart mobile devices for navigation. First, a mobile device (client application) queries the server for the 3D map of the environment using the unqiue identifier associated with that environment, i.e., `Key_point ID`. After retrieving the corresponding keypoint descriptor, it is extracted by the client. The client then scans the surrounding environment, matching the current keypoints with the retrieved ones to localize its position. This way, the client realizes its location within the 3D map.

For navigation, the client relies on the matched current coordinates and the destination name that is given as a location tag within the keypoint descriptor. Then the client uses the shortest path information coming from the keypoint descriptor, and displays AR-based direction markers.

C. Prototype implementation details

We have built our navigation solution leveraging ARKit, Apple's augmented reality SDK, which is essential and appealing for many iOS-based AR systems as it provides different customizations and open-source pre-built functions. Unlike other AR SDKs, ARKit provides a low power, cost-efficient, and jitter-free solution. The framework consists of three major modules: ARSceneKit, ARWorldMap, and Unity plugin for ARKit. ARSceneKit facilitates adding 3D objects in an AR environment. AR objects are taken from Unity's Asset Store, which in our application are AR Texts, being used to anchor various location names with the environment, e.g., different office names, hall entries, etc. ARSceneKit creates an AR experience through merging these 3D objects with the camera view of the real world [31]. The AR experience is achieved by rendering these 3D objects over the live feed from the camera. ARKit then matches the real-world moments of the device with the SceneKit [32] camera.

The mapping module which is used in the scanning of the environment and creating a 3D map of it, employs ARWorldMap. This creates a keypoint point cloud, embedded in `ARWorldMap_file`. The coordinates of the next keypoint is created based on the relative location of the previously generated coordinates. Therefore, for faster scanning of the environment the camera should be positioned in a way that previously generated keypoints and location tags are visible, as confirmed by our experiments.

In the localization phase, ARKit tracking modules are employed to find the location of a mobile device by matching the observed features with the keypoint descriptor that is sent by the server. In this process, G2O module [33] is used for optimizing nonlinear least squares problems that can be embedded as a graph or as a hypergraph.

We have deployed the shared remote server on an NGINX server [34] that is used to store and retrieve the 3D map constructions along with the shortest paths between location tags. Samples of 3D maps from our test runs for a two-story building is visualized and depicted in Figure 3. The 3D reconstruction of this map is accomplished using an open-source monocular visual SLAM framework known as OpenVSLAM [35]. The visualization of the 3D map is done using a rapid development library for abstracting video input, known as Pangolin [36], along with G2O for graph optimizations.

Apple's Socket and Socket Streams modules [37] are used in both the storage of keypoint descriptors in the server (phase 1), and the retrieval of them by the client applications (phase 2). Using HTTPS through these modules, the client-server communication channel is secured. In addition, since the client application queries the server by only providing the corresponding keypoint identifier of the environment, the exact location is not revealed to the server. This provides some level of privacy for the application users.

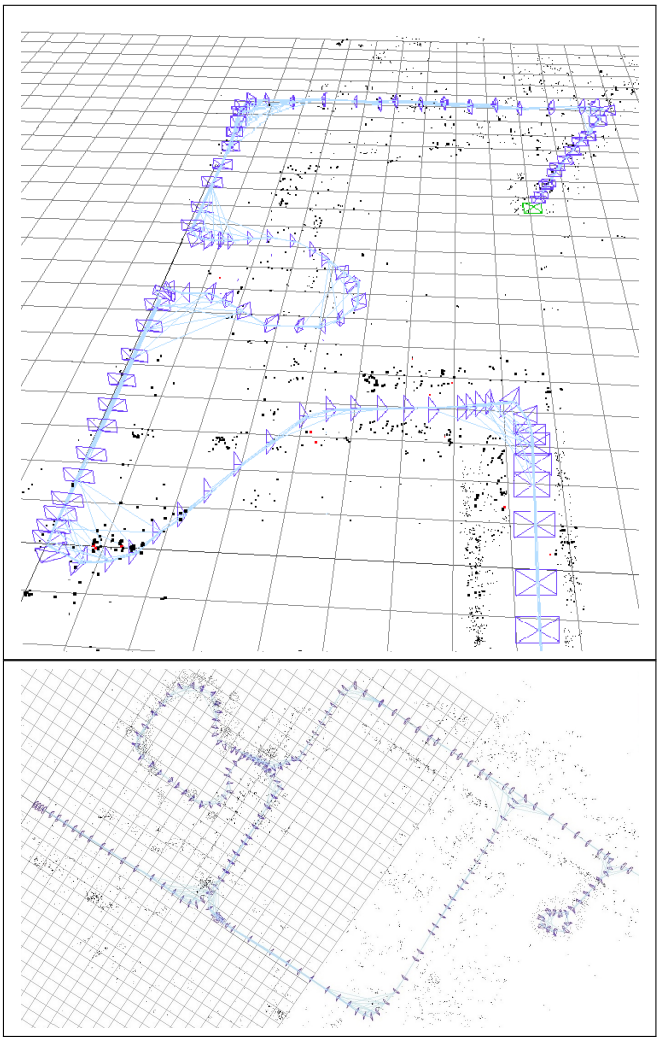

Fig. 3. A sample 3D map generated by the SLAM server.

Our test runs demonstrate that in the localization phase, it takes 3-4 seconds to load and place the AR location tags in environments with sufficient amount of light. In darker locations, initial localization takes 5-7 seconds. In a completely dark environment, however, localization is not successful. Once loaded, the location tags remain in their designated position even with oriented, rotated and fast movements of the camera. The algorithm to find the shortest path and generate corresponding relative directional markers on the mobile device works in realtime. Figure 4 shows sample screenshots of the mobile application at the time of navigation.

One of the limitation that we have found during system tests, deals with the plain, solid-colored walls while constructing the map. This is quite common in any feature extracting algorithm, since the featured keypoints are usually the corners, edges and sudden color intensity difference in an image. Having monochrome walls makes it difficult for the algorithm to extract feature points.

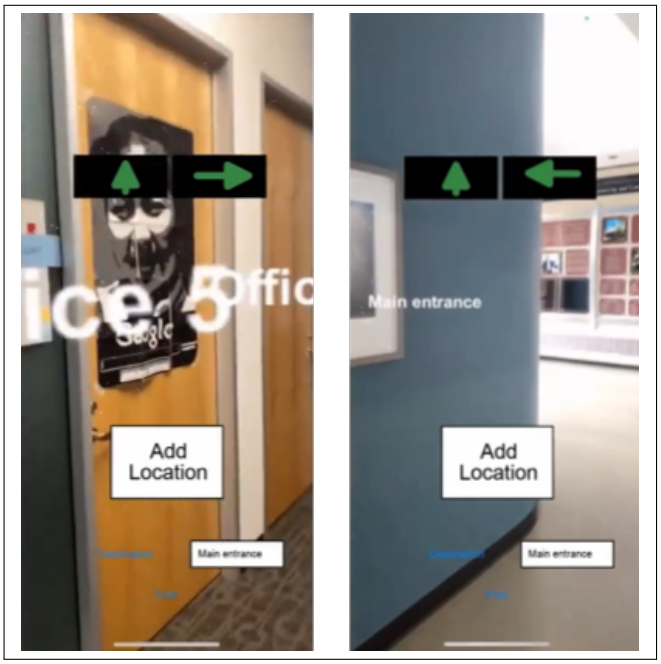

Fig. 4. Sample screenshots of mobile application helping to navigate the user.

IV. Related work

When it comes to vision-based indoor navigation systems, there are two types of methods to implement an IPS/INS: Marker and Markerless [38]. Generally, this marker is a unique, distinguished landmark identifier which can be easily detected by camera. When a system recognizes these markers, it adds them to the database with relevant information such as location tags, current coordinate information of the marker, etc. Kim et al. [5] use unique markers distinguished from the background. The remote server receives a sequence of image frames from a hand-held camera, where these markers are identified and the associated location information is retrieved. Based on this information, AR markers are generated on a head mounted display. Raj et al. [39] use QR codes as markers. Their system has the advantage of providing additional privacy, as all computations are performed locally. However, a common disadvantage of using a visual marker based INS is the lack of realtime positioning feedback of the device. De Oliveria et al. [40] propose custom AR fiducial markers and use bluetooth beacons to provide INS for wheelchair users. This implementation blends visual markers with the beacon technology to get the realtime information through beacon, and accommodates precise location tracking through fiducial markers.

Among markerless implementations, Faragher et al. [41] have deployed a radio SLAM for IPS, using a point cloud generated from the data collected by various smartphone sensors using distributed particle simultaneous localization and mapping (DPSLAM) algorithm [42]. The main advantage of this approach is the lack of dependency to any prior knowledge about the floor plans. Their proposed system includes various inertial measurement unit sensors, hooked to the user body. DPSLAM uses an efficient database to maintain the previous state histories. The particle cloud states are created using the outdoor GNSS signals before a user enters the building via signal fingerprinting from a prior map, in a region where such map is available. GNSS is also used to detect the initial location of the smartphone before the user enters the building.

We observe a common set of issues in these implementations. In wireless markerless INS, one of the common problems is to figure out the initial location of the device. For visual marker-based implementations, the issue is the lack of realtime feedback regarding the device's current location. Additionally, scanning the fiducial markers along the way is a usability concern. Our proposed system overcomes these issues by successfully localizing the initial position without the need of any visual markers. Furthermore, the proposed INS is realtime and can be used by multiple clients simultaneously.

V. Future work and conclusion

This paper proposes an indoor navigation system that is markerless and vision-based, relying on AR SDKs of modern smart mobile devices, in particular ARKit for iOS platform. The proposed system consists of a shared remote server to store and retrieve the 3D maps of indoor environments. In the initial phase, the mobile device (client) scans environment using the built-in camera and constructs the 3D map using ARWorldMap, as well as the shortest paths between user-defined location tags. These information are then uploaded to the server, which accommodates indoor navigation for arbitrary mobile devices in a later phase. In the navigation phase, a mobile device can help user navigate the path using directional markers, after receiving the 3D map of the environment and the shortest paths between location tags, followed by locating the device within that environment. The observed limitation of the proposed system is the reconstruction of the environment with physical structures without changes in the color intensities.

Although the application is currently limited to iOS platform, the Unity plugin makes it possible to extend the application to other platforms as it supports both Android and iOS. In this regard, we are planning to extend the implementation to other platforms, in particular Android.

Another potential future work is to study the ways to integrate the proposed indoor navigation system with existing mature outdoor navigation systems, with the goal of providing a seamless experience to the end user.

References

[1] H. Koyuncu and S. H. Yang, "A survey of indoor positioning and object locating systems," *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 5, pp. 121–128, 2010.

[2] A. Butz, J. Baus, A. Krüger, and M. Lohse, "A hybrid indoor navigation system," in *Proceedings of the 6th international conference on Intelligent user interfaces*, 2001, pp. 25–32.

[3] Z. Jianyong, L. Haiyong, C. Zili, and L. Zhaohui, "Rssi based bluetooth low energy indoor positioning," in *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2014, pp. 526–533.

[4] M. Jain, R. C. Rahul, and S. Tolety, "A study on indoor navigation techniques using smartphones," in *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2013, pp. 1113–1118.

[5] J. Kim and H. Jun, "Vision-based location positioning using augmented reality for indoor navigation," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 954–962, 2008.

[6] C. Koch, M. Neges, M. König, and M. Abramovici, "Natural markers for augmented reality-based indoor navigation and facility maintenance," *Automation in Construction*, vol. 48, pp. 18–30, 2014.

[7] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[8] S. H.-W. Chuah, "Why and who will adopt extended reality technology? literature review, synthesis, and future research agenda," *Literature Review, Synthesis, and Future Research Agenda (December 13, 2018)*, 2018.

[9] J. Johnson, "The master key: L. frank baum envisions augmented reality glasses in 1901," *Mote & Beam*, 2012.

[10] W. Wang, "Understanding augmented reality and arkit," in *Beginning ARKit for iPhone and iPad*. Springer, 2018, pp. 1–17.

[11] T. Feigl, A. Porada, S. Steiner, C. Löffler, C. Mutschler, and M. Philippsen, "Localization limitations of arcore, arkit, and hololens in dynamic large-scale industry environments." in *VISIGRAPP (1: GRAPP)*, 2020, pp. 307–318.

[12] C. R. Ehrlich and J. Blankenbach, "Pedestrian localisation inside buildings based on multi-sensor smartphones," in *2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)*. IEEE, 2018, pp. 1–10.

[13] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 3562–3568.

[14] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.

[15] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[16] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[17] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. Ieee, 1999, pp. 1150–1157.

[18] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[19] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[20] T. Schöps, J. Engel, and D. Cremers, "Semi-dense visual odometry for ar on a smartphone," in *2014 IEEE international symposium on mixed and augmented reality (ISMAR)*. IEEE, 2014, pp. 145–150.

[21] "ARWorldMap," https://rb.gy/q7ozwa, 2020.

[22] H. Huang and G. Gartner, "A survey of mobile indoor navigation systems," in *Cartography in Central and Eastern Europe*. Springer, 2009, pp. 305–319.

[23] U. Steinhoff and B. Schiele, "Dead reckoning from the pocket-an experimental study," in *2010 IEEE international conference on pervasive computing and communications (PerCom)*. IEEE, 2010, pp. 162–170.

[24] F. Thomas and L. Ros, "Revisiting trilateration for robot localization," *IEEE Transactions on robotics*, vol. 21, no. 1, pp. 93–101, 2005.

[25] J. B. Bancroft, D. Garrett, and G. Lachapelle, "Activity and environment classification using foot mounted navigation sensors," in *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2012, pp. 1–10.

[26] C. Yang and H.-R. Shao, "Wifi-based indoor positioning," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 150–157, 2015.

[27] A. M. Nor and E. M. Mohamed, "Li-fi positioning for efficient millimeter wave beamforming training in indoor environment," *Mobile Networks and Applications*, vol. 24, no. 2, pp. 517–531, 2019.

[28] B. Ozdenizci, K. Ok, V. Coskun, and M. N. Aydin, "Development of an indoor navigation system using nfc technology," in *2011 Fourth International Conference on Information and Computing*. IEEE, 2011, pp. 11–14.

[29] C. Lee, Y. Chang, G. Park, J. Ryu, S.-G. Jeong, S. Park, J. W. Park, H. C. Lee, K.-s. Hong, and M. H. Lee, "Indoor positioning system based on incident angles of infrared emitters," in *30th Annual Conference of IEEE Industrial Electronics Society, 2004. IECON 2004*, vol. 3. IEEE, 2004, pp. 2218–2222.

[30] A. Bajpai and S. Amir-Mohammadian, "SLAM-based-Indoor-Navigation," https://github.com/Akbonline/SLAM-based-Indoor-Navigation, 2021.

[31] "ARSCNView," https://developer.apple.com/documentation/arkit/arscnview, 2020.

[32] "SceneKit," https://developer.apple.com/documentation/scenekit/, 2020.

[33] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

[34] W. Reese, "Nginx: the high-performance web server and reverse proxy," *Linux Journal*, vol. 2008, no. 173, p. 2, 2008.

[35] S. Sumikura, M. Shibuya, and K. Sakurada, "Openvslam: a versatile visual slam framework," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2292–2295.

[36] S. Lovegrove, "Pangolin," https://github.com/stevenlovegrove/Pangolin, 2020.

[37] "Using Sockets and Socket Streams," https://rb.gy/hklbcjl, 2013.

[38] J. Cheng, K. Chen, and W. Chen, "Comparison of marker-based ar and marker-less ar: a case study on indoor decoration system," in *Lean and Computing in Construction Congress (LC3): Proceedings of the Joint Conference on Computing in Construction (JC3)*, 2017, pp. 483–490.

[39] C. R. Raj, S. Tolety, and C. Immaculate, "Qr code based navigation system for closed building using smart phones," in *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*. IEEE, 2013, pp. 641–644.

[40] L. C. De Oliveira, A. O. Andrade, E. C. De Oliveira, A. Soares, A. Cardoso, and E. Lamounier, "Indoor navigation with mobile augmented reality and beacon technology for wheelchair users," in *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*. IEEE, 2017, pp. 37–40.

[41] R. Faragher, C. Sarno, and M. Newman, "Opportunistic radio slam for indoor navigation using smartphone sensors," in *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*. IEEE, 2012, pp. 120–128.

[42] A. I. Eliazar and R. Parr, "Dp-slam 2.0," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 2. IEEE, 2004, pp. 1314–1320.