Outdoor Navigation System by AR

Ryohei Hashimoto[1,*] and Michael Cohen[1,*]

[1]Spatial Media Group, Computer Arts Lab; University of Aizu; Aizu-Wakamatsu, Fukushima 965-8580; Japan

Abstract. Mobile map applications such as Google Maps often don't provide detailed information about facility areas such as amusement parks and university campuses. In addition, there are some people who cannot reach their destination just by reading a flat map. Therefore, in this research, we have developed an AR (Augmented Reality) navigation application for facilities to solve these problems. In addition, by using Kalman filtering to estimate user position, we could improve the accuracy of AR objects display.

1 Introduction

People increasingly use smartphone navigation apps instead of paper maps when traveling. Many of these cartographic apps not only allow users to check their current location using the GNSS (Global Navigation Satellite System, such as GPS) installed in their smartphones, but also have way-finding functions, guidance to selected destinations.

However, navigation of outdoor sites using such map apps has the following problems:

1. Detailed information needed to get to destination is often not included.

   In most cases, a user's destination is not the site itself but a facility located within the site. However, these map apps often do not provide information about on-site facilities. In that case, it is necessary to refer to a separate, detailed map dedicated to the site.

2. Users should have the ability to read maps.

   These map apps still require users to be able to read maps. According to [1], in order to read maps, it is necessary to have ability to switch from subjective view to objective view and an ability to rotate perspective mentally. There are individual differences in these abilities. It is not yet clear where this difference occurs, but this fact indicates that some people are not good at reading maps. For them, it can be difficult to reach a target facility at a site that is visited for the first time.

According to [2], the number of people who get lost can decrease dramatically by preparing a guide using photographs and sentences based on visual information from landmarks for those who cannot read maps well. Such users are more likely to be able to reach their destination by presenting information related to their point of view. Since AR (Augmented Reality) augments a user's perspective, AR can be one of the best guides for navigation. Therefore, in this paper, we propose an AR navigation system that can be used at outdoor sites. This prototype is a navigation system for outdoor sites such as zoos, theme parks, and university campuses, and users can smoothly go to their destinations by receiving AR

Figure 1: AR navigation system for outdoor sites

*e-mail: {s1250016,mcohen}@u-aizu.ac.jp

navigation suggestions.

An outdoor site AR navigation system for smartphones developed to solve these two problems is reported. User position estimation is important to realize AR navigation. Therefore, this system is equipped with a Kalman filter that integrates data from inertial sensors in addition to the GNSS position sensing.

2 Navigation apps which are published currently

When navigating using a smartphone, it is common to use map apps such as Google Maps [3] by Google and Maps [4] by Apple. However, navigation is often possible only along public roads with these apps. That's because APIs such as the Google Maps Platform [5] don't provide detailed information on premises such as theme parks and campuses.

In contrast, there exist navigation apps specialized for particular facilities. Examples of these include the Tokyo Disney Resort app [6] and Tokyo Parks Navi [7]. Unlike map apps such as Google Maps, these apps specialize in information regarding particular premises, so it is possible to navigate even in nonpublic areas where navigation is not supported with general map apps. However, these apps are often published by each facility, so there is a disadvantage that users must switch applications for each facility. Tokyo Parks Navi handles information for multiple facilities in Tokyo such as Ueno Zoo and Tokyo Sea Life Park, but as its name indicates, the set of facilities that it supports is limited to Tokyo.

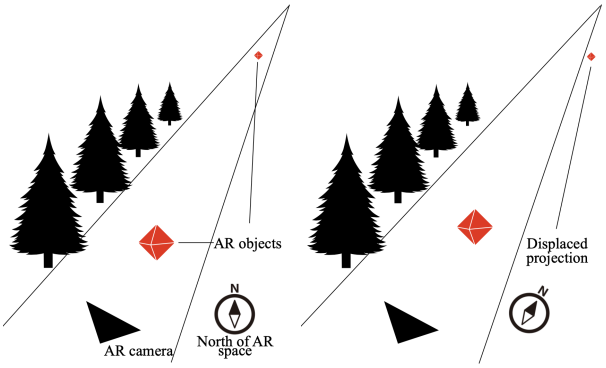
In summary, there is the problem that there exist no general-purpose navigation apps for facilities. This project comprises research and development aimed at addressing this problem.

3 Development of in-facility AR navigation system

3.1 Methods of Navigation

The usual contemporary navigation method is to draw one's current location obtained by GNSS and a suggested route to the destination obtained by API access on a two-dimensional map displayed on a smartphone. This is the technique used by most map and navigation apps. However, this method requires users to be able to read a map. Therefore, this method is not suitable for people who are not good at reading maps.

To solve this problem, a navigation system using AR technology is proposed. The first possible method for this is to use GNSS and a magnetometer (electronic compass) to place navigation objects displayed in AR space. However, accuracy of GNSS varies depending upon the environment in which it is used. Therefore, research on



(a) Case when north of the AR space and north of the real space are aligned

(b) Case when north of AR space does not match north of real space

Figure 2: Impact of magnetometer error in AR space

AR navigation that does not depend on GNSS is being conducted. These studies explore a method of determining current location with markers [8] and a method of using landmark confirmation at intersections [9]. In addition, VPS (Visual Positioning Service) using imagery from mobile device cameras and machine learning is also being researched [10]. However, these methods require time and effort, such as arranging markers in various places and recording landmark information at each intersection. With navigation using GNSS and a magnetometer, it is possible to navigate with only the intrinsic sensors installed in a smartphone. If the environment is good, it is possible to obtain location accuracy within about 5 m. Since our system is intended for use in situations where the GNSS reception environment is relatively good, such as in amusement parks and universities, this project uses GNSS and magnetometer for AR navigation.

There are several ways to configure AR navigation using GNSS and an magnetometer, but each method requires AR space to be aligned with the real space. A magnetometer is used for matching azimuth, but it cannot be perfectly matched due to error. If the angle is off, as shown in Fig. 2b, there is a problem that an AR object could be displayed in the wrong position. In order to solve this problem, this project adopted a method of rendering a map in AR space and mapping it into real space. After preliminary automatic alignment with the magnetometer, the user manually adjusts the azimuth. When adjusting, the overlay can be calibrated by aligning a road object displayed in AR with the corresponding roadway in real space.

3.2 Map services

There are two GIS (Geographic Information System) services that can be readily used to display maps in AR: Google Maps and Mapbox [11]. In our proof-of-concept, such service is used to draw a map and acquire a route. There are advantages and disadvantages regarding these two services, and it is not easy to choose which to use. For

example, consider how accurate the walking navigation route in the facility is, which depends on how many walkways in the large facilities exist in the map database. For Tokyo Disney Resort, both Google Maps and Mapbox have pedestrian paths, and navigation routing is accurate. However, at Nasu Highland Park, Google Maps does not have a walking path to some facilities, and it is not possible to determine an optimal navigation route. Mapbox registers a lot of walking paths in the facility, so it is possible to determine an appropriate route. On the other hand, at the University of Aizu, Google Maps has some walking paths, but Mapbox may output routes that require a round-about detour to a destination, due to the small number of cartographic walking paths it has registered for the premises.

In this project, Unity is used for development. Mapbox provides an SDK for Unity and has abundant examples for AR sample scenes, so this project uses the Mapbox service. (Google maps also has an SDK for Unity, but its development support doesn't include AR sample scenes.)

3.3 Development environment

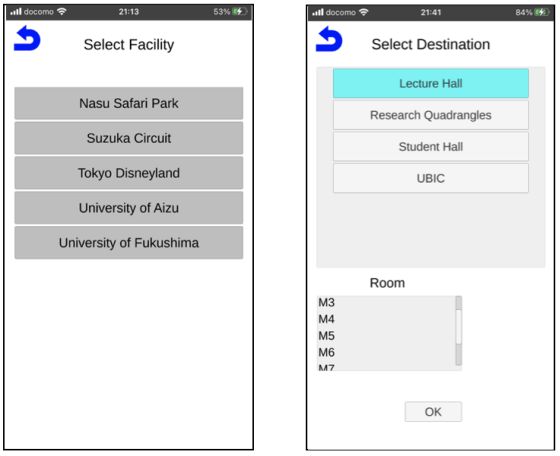
Research and development that focuses on such issues are discussed in [12], which project was developed in the native environments of both iOS and Android, and which was intended to reduce development time by matching corresponding program structures. In the project described in this paper, it is possible to develop a single set of source files compatible with multiple platforms by using Unity as a cross-platform development environment. This can significantly reduce programming time compared to developing platform-specifically, and it will be easier to manage future extensions.

3.4 Functions

3.4.1 Facility and destination selection

This system expects a user to know the name of a destination in advance. First, the user selects on the facility selection view the facility to which the destination belongs (Fig. 3a). When a facility is selected by the user, the system displays candidate destinations belonging to that facility. In this project, data about facilities has been stored in advance for experiments, but it is preferable to implement an API that returns facility information dynamically using the facility name as a database query. In addition, a system that allows facility managers to edit facility information is also desirable.

Next, the user selects the name of the place to go to on the destination selection view (Fig. 3b). If the destination is a building and has a room inside, it is also possible to indicate the room information included in the destination by tapping the destination name.



(a) Facility Selection (b) Destination Selection

Figure 3: Facility and Destination Selection views

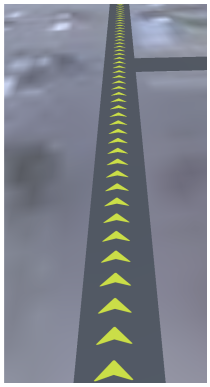
3.4.2 AR navigation to the destination

After a destination is selected by the user, the system accesses the Mapbox API using the user's current location and destination coordinates as a query to calculate a route. Once the route is obtained, navigation objects are placed along it. There are two types of navigation objects: route objects which represent walking routes with arrows (Fig. 4a), and waypoint objects located at each waypoint such as intersections and T-junctions (Fig. 4b). The user can confirm a walking route by extended route object, and can confirm each waypoint by each waypoint object. We judged that it would be excessive to have waypoints for all intersections and T-junctions, so they are only located at right or left turn-points. In addition, to enable navigation even when the user is not looking at the smartphone, a function that displays a recorded navigation voice guidance according to the type of waypoint as the user approaches each waypoint object is implemented. The current types of navigation voice commands are as follow.

- Departure point (start)
  - Walk north.
  - Walk south.
  - Walk west.
  - Walk east.
- Intersection (turn-point)
  - Turn left.
  - Turn right.
- Destination (goal)
  - You've arrived at destination.

For waypoint objects, start and destination markers are colored red, and intermediate markers are colored green.

At this time, for departure point, intermediate directions are not supported. The navigation voice is currently rendered diotically, but it is possible to make the system so that the sound is heard from the direction of the next waypoint, like Microsoft Soundscape [13].



(a) Route object (series of chevron arrows)

(b) Waypoint objects (green or red markers)

Figure 4: Navigation AR overlay objects

3.4.3 Matching of AR space and real space

Before starting AR navigation, the AR space must be aligned with the real space. The matching operation used by our system is to match the north of the AR space with the north of the real space, as described in §3.1. By using an magnetometer, two values representing north (magnetic north and true north) are obtained. In this project, true north is used. This process can be divided into two phases: automatic adjustment that aligns a certain azimuth with an magnetometer, and manual adjustment that brings an error closer to zero with human calibration. The automatic adjustment algorithm follows, where the project coordinate system uses a left-handed arrangement in which the x-axis is lateral sway, the y-axis is vertical heave (gravitational up), and the z-axis is longitudinal surge. Fig. 5 is a diagram of vectors related to this algorithm.

Step 1. By subtracting y-axis rotation amount β of AR camera in AR space from current smartphone azimuth α in real space, real space orientation of z-axis of AR space θ_0 is obtained. That is the offset rotation amount of AR space.

Step 2. Counter-rotate the map in AR space around the y-axis by offsetting by $-\theta_0$.

3.5 Handling of AR objects

3.5.1 AR system

Currently, there are basically two types of AR: marker-based AR and markerless AR. In the former method, a specific marker is prepared and corresponding AR object

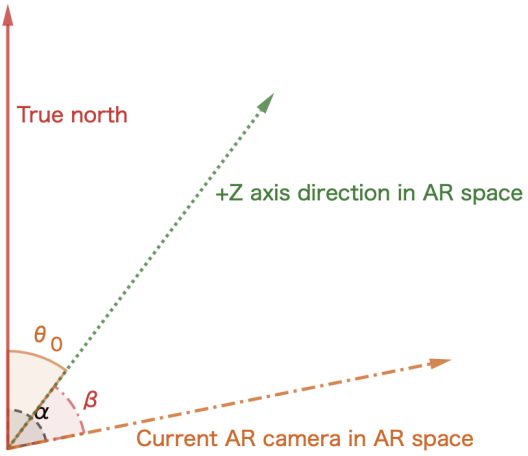


Figure 5: Vector diagram of xz plane view (plan perspective) when aligning north direction, where β is y-axis rotation amount, α is current smartphone azimuth, θ_0 is real space orientation of z-axis of AR space

is placed at the marker; in the latter method, walls or the ground are detected by image processing technology and AR objects are placed along their planes. Unity's AR Foundation supports both methods, but our system uses markerless AR because it detects the ground (plane) and places AR objects upon it.

The AR system detects the ground when the user aims the camera at the ground. When the system accesses Mapbox's API to obtain a map and route data, some AR objects (a map around user's current location, road objects, and navigation objects) are generated in the AR space. At this time, the user's current location in the real space is correctly projected into the AR space. If a plane map is drawn on a view through AR camera (device camera), it would be difficult to see the ground in the real space. For that reason, only road objects and navigation objects are drawn on the AR scene. By detecting the ground with the AR system, AR objects (road objects and navigation objects) are aligned with the ground in real space. Since AR objects are placed on the plane map in the AR space, AR objects won't be displayed correctly in the real space if the pedestrian path in the real space is a slope, as seen in Fig. 6. To solve this problem, the heights of the AR objects are adjusted each time a new plane is detected. It is desirable to encourage users to re-detect the ground as the height of the ground changes, but we defer such issues for now.

3.5.2 Updating coordinates of AR camera object

Unity provides an API to access location information. This API uses the update cycle of location information and the accuracy of location information as parameters. If these values are set to high level (shorter update cycle and higher accuracy), more accurate position information can be obtained, but more battery energy is consumed.

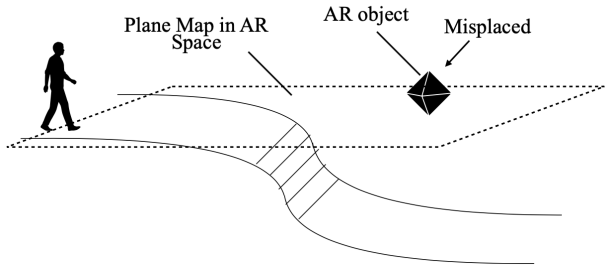


Figure 6: Misplacement of AR objects with height gradient

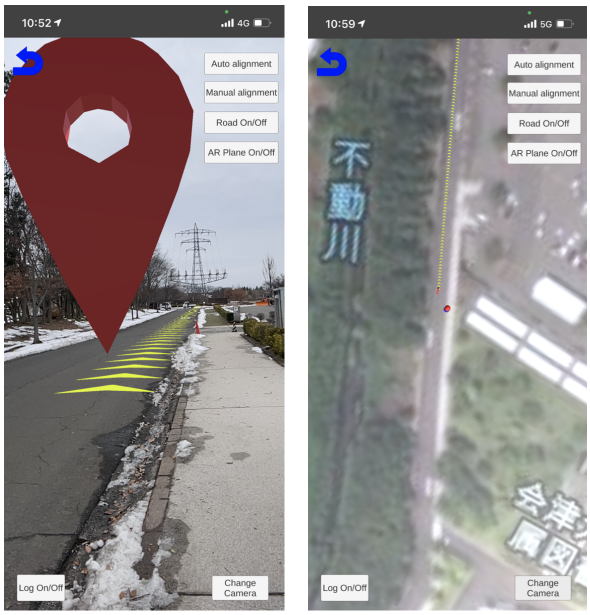
However, in this system, which needs to constantly map a AR space to the real space, the update cycle should be as short as possible, and the position accuracy should be set to higher accuracy. Therefore, in this system, the update cycle is set to 1 s and the accuracy is set to the maximum (an error of about 5 m). The shortest update cycle that can be set is 0.5 s, but with that setting, the battery drains significantly because the AR function is overclocked.

When updating a user's position in AR space, coordinates of the AR camera are changed directly. Assuming that directions of an AR space and real space are aligned, by mapping user's current location (AR camera in the AR space) correctly on the map of the AR space, AR objects are displayed correctly on the "see-through" device (Fig. 7). However, in Unity's AR Foundation, coordinates of the AR camera object are measured by dead reckoning, and it is impossible to change coordinates of the AR camera object directly. In Unity, a hierarchy of objects is comprised of parent-child relationship, and when a `Transform` (coordinates, orientation, scale) of a parent object changes, the child object inherits such relative parameters. Therefore, coordinates of the AR camera object are changed by preparing an object attached in the scene hierarchy as the parent of the AR camera object and changing coordinates of the parent object.

Kalman filtering (described below) is processed at update timing of the device position information, and position of the AR camera at that time is synchronously calculated. By moving the parent object of the AR camera object so that the AR camera object overlays its viewport, coordinates of the AR camera in the AR space are dynamically updated.

3.6 System configuration

A schematic diagram of this system is shown in Fig. 9. The client app is composed with Maps SDK for Unity, which is an SDK for Unity of Mapbox, and AR Foundation, which is an AR platform of Unity. The server side is mainly composed of Mapbox services. There are two types of services to use: maps service, which serves map data, and navigation service, which serves routes. Static Images API



(a) View through device camera (AR Camera)

(b) View using imagery satellite in AR space

Figure 7: Images showing that user's current location is plotted on a map in AR space and AR objects are displayed correctly.

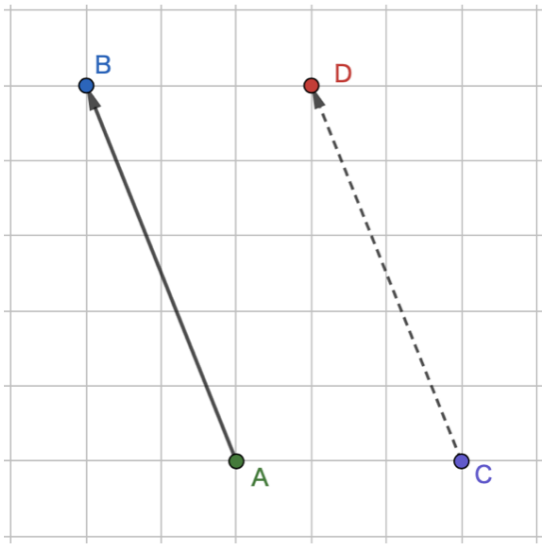


Figure 8: Motion of AR camera by parent object, where A is parent of AR camera, B is parent of AR camera (after movement), C is AR camera, D is calculated position (destination of AR camera)

and Vector tiles API are used, which are included in maps service, and Directions API is used, which is included in navigation service. Sensor data includes inertial sensors, GNSS, and magnetometer. Since this app is used outdoors, a cellular network is usually used for communication with cloud-hosted server. A device must be online to get map

and route data, but since system doesn't access the internet during navigation, it can be used offline after caching way-finding information.

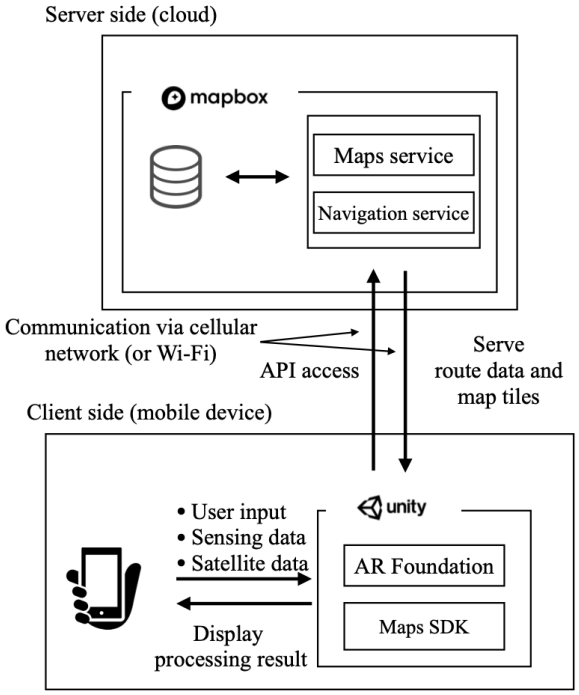


Figure 9: System overview

4 User position estimation using Kalman filtering

If the location estimation is too far from the true value, an AR object cannot be placed in its proper position, and navigation becomes impossible. Therefore, this system aims to improve estimation accuracy of user position by implementing a Kalman filter. A Kalman filter is a filter for estimating system status using noisy (inaccurate) observations by minimizing statistically estimated error (mean square deviation). Several studies have been conducted on the improvement of GNSS accuracy by Kalman filter [14] [15]. In this project, since movement information of the AR camera measured by inertial sensor of the smartphone can be used, a Kalman filter that integrates this data and GNSS data was implemented.

4.1 Time update formula

Estimated state $\boldsymbol{x}(k)$ is defined as

$$\boldsymbol{x}(k) = \begin{pmatrix} x(k) \\ z(k) \end{pmatrix}, \tag{1}$$

where $\boldsymbol{x}(k)$ is the true state at time k and $x(k)$, $z(k)$ are the true x and z coordinates of a user in Unity space at time k. In the following, the relationship between the output of each sensor and the state $\boldsymbol{x}$ is derived.

First, let $\boldsymbol{u}(k)$ be true distance traveled by the AR camera from time k to time $k+1$, as measured by inertial sensors (accelerometer and gyro sensor) of the smartphone IMU (Inertial Measurement Unit). Then, subsequent noiseless position can be expressed as

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \boldsymbol{u}(k). \tag{2}$$

The following equation of perturbed state is obtained from extension of Equation (2):

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \boldsymbol{u}(k) + \boldsymbol{v}(k), \tag{3}$$

where $\boldsymbol{v}(k)$ is the system noise, including centered noise of the inertial sensor having mean value vector $\mathbf{0}$ (zero vector) and covariance matrix $\boldsymbol{Q}$.

Next, since the value output by GNSS is the absolute location of the smartphone on the earth, it can be associated with coordinates in Unity space and designated $\boldsymbol{y}(k)$. Then, the following observation equation is obtained:

$$\boldsymbol{y}(k) = \boldsymbol{x}(k) + \boldsymbol{w}(k), \tag{4}$$

where $\boldsymbol{w}(k)$ is the observation noise of GNSS, having mean value vector $\mathbf{0}$ and covariance matrix $\boldsymbol{R}$.

When the Kalman filter is programmed using equations (3) and (4) as the state space model of the system, the following time-updated compound equation of the Kalman filter is obtained:

$$\begin{aligned}
&\hat{\boldsymbol{x}}(k|k-1) = \hat{\boldsymbol{x}}(k-1|k-1) + \boldsymbol{u}(k-1), \\
&\boldsymbol{P}(k|k-1) = \boldsymbol{P}(k-1|k-1) + \boldsymbol{Q}, \\
&\boldsymbol{G}(k) = \boldsymbol{P}(k|k-1)(\boldsymbol{P}(k|k-1) + \boldsymbol{R})^{-1}, \\
&\hat{\boldsymbol{x}}(k|k) = \hat{\boldsymbol{x}}(k|k-1) + \boldsymbol{G}(k)(\boldsymbol{y}(k) - \hat{\boldsymbol{x}}(k|k-1)), \\
&\boldsymbol{P}(k|k) = (\boldsymbol{I} - \boldsymbol{G}(k))\boldsymbol{P}(k|k-1),
\end{aligned} \tag{5}$$

where $\hat{\boldsymbol{x}}(k|k-1)$ is *a priori* state estimate value, $\boldsymbol{P}(k|k-1)$ is *a priori* estimate covariance matrix, $0 \leq \boldsymbol{G}(k) \leq 1$ is the Kalman gain, $\hat{\boldsymbol{x}}(k|k)$ is *a posteriori* state estimate value, $\boldsymbol{P}(k|k)$ is *a posteriori* estimate covariance matrix, and $\boldsymbol{I}$ is the identity matrix.

Initial values of state estimate value and the estimate covariance matrix are expressed as

$$\begin{aligned}
\hat{\boldsymbol{x}}(0|0) &= \boldsymbol{x_0}, \\
\boldsymbol{P}(0|0) &= \gamma \boldsymbol{I},
\end{aligned} \tag{6}$$

where $\boldsymbol{x}(0|0)$ is the initial coordinates of the AR camera, to be determined by the user before starting of AR navigation, and γ is one of the calibration parameters, within range 0–1000.

4.2 Experiment and result

Adjustment parameter values of this Kalman filter were determined heuristically by walking experiments as follows.

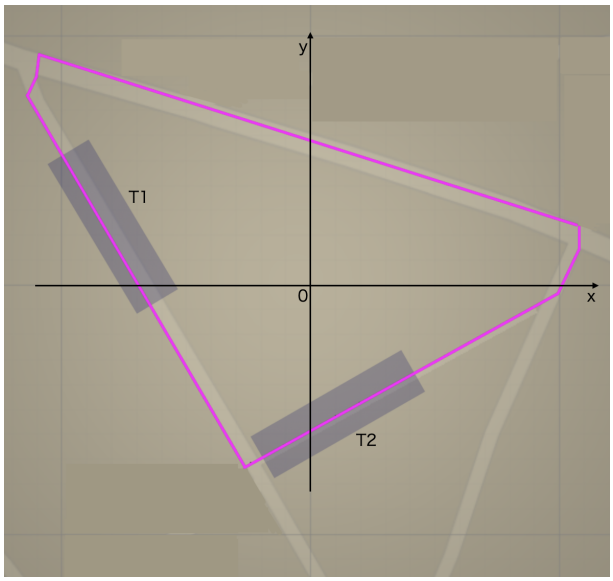


Figure 10: Walking route in verification experiment

$$\begin{aligned} \boldsymbol{Q} &= \sigma_v^2 \delta_{ij} \\ \boldsymbol{R} &= \sigma_w^2 \delta_{ij} \\ \gamma &= 1.0 \end{aligned} \tag{7}$$

Here, $\sigma_v = 0.2$, $\sigma_w = A\eta$, A is the horizontal accuracy of GNSS (unit is meter), and η is the adjustment magnification of σ_w, calculated as

$$\eta = \begin{cases} 0.3 & (\text{accuracy} < 10) \\ 0.5 & (10 \leq \text{accuracy} < 30) \\ 3 & (30 \leq \text{accuracy}), \end{cases} \tag{8}$$

and δ_{ij} is the Kronecker delta. The adjustment magnification is changed according to the accuracy of GNSS to improve filtering performance. For example, when the accuracy is 5 m, we want the system to strongly trust observed values, but at 10 m, we want the system to trust estimated values somewhat more. However, if the accuracy of GNSS is used as observation noise as it is, the observed values will be strongly trusted even at 10 m.

No method has been discovered to automatically determine these parameters. This time, the approximate optimum value was derived by repeating walking experiments. Although these are defined as above in this system, it is acknowledged that there is an error from the optimum value in reality.

A verification experiment was conducted to confirm the effectiveness of the above settings. The walking route is shown in Fig. 10. Since the walking route used in the experiment had relatively good environment for GNSS reception, objects imitating tunnels were placed in the AR space to intentionally reduce GNSS reception intensity. T1 and T2 in Fig. 10 are simulated tunnel objects. When the tracking device enters T1 and T2, the accuracy becomes 10 m and 30 m, respectively. The original GNSS location information was used to determine if the device has entered tunnels.

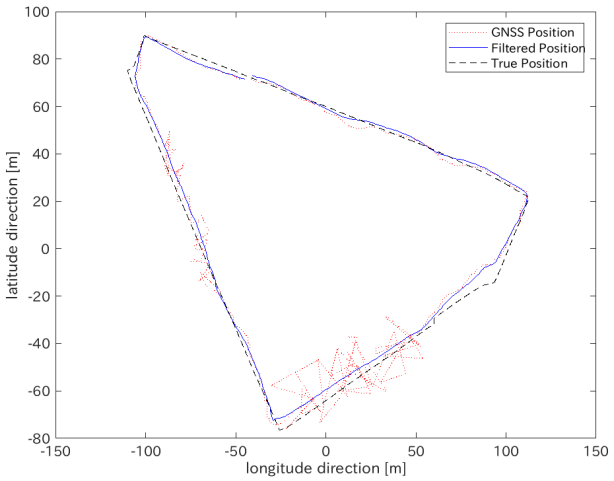


Figure 11: Verification experiment results

The result of the verification experiment is shown in Fig. 11. As soon as the device enters T1 or T2, the GNSS data starts to include a lot of noise, but the filtered data contains negligible noise.

5 Conclusion

We configured a navigation app for location-based facilities and developed an AR navigation function. In addition, a Kalman filter was implemented to enable navigation that is not easily degradable by inaccuracy of GNSS.

For this project, Mapbox's API was used for route acquisition, but depending on the facility, sufficient navigation may not be possible, so in such cases it would be necessary to build a route acquisition and registration system. In addition, a cartographic management system for facility managers would also be required. In the future, we would like to develop a system that includes these functions and have facility users try it to conform usability. It is also conceivable to build a VPS specialized for a facility. Furthermore, we would like to consider an AR eyewear version of this system.

References

[1] *Bad at Reading Maps? Maybe Your Brain Just Needs Better Maps | WIRED*, https://www.wired.com/2013/11/map-sense/, (Accessed on 01/08/2021)

[2] *"I can't read the map!" Lamentation of those who get lost | AERA dot. | Toyo Keizai Online | New standard for economic news*, https://toyokeizai.net/articles/-/158329, (Accessed on 01/08/2021)

[3] *Google Maps*, https://www.google.com/maps/, (Accessed on 10/31/2020)

[4] *Maps — Apple*, https://www.apple.com/maps/, (Accessed on 10/31/2020)

[5] *Geo-location APIs | Google Maps Platform | Google Cloud*, https://cloud.google.com/maps-platform, (Accessed on 10/31/2020)

[6] *Tokyo Disney Resort App*, https://www.tokyodisneyresort.jp/topics/tdrapp.html, (Accessed on 10/31/2020)

[7] *Tokyo Parks Navi*, https://www.tokyo-park.or.jp/special/tokyoparksnavi/index.html, (Accessed on 10/31/2020)

[8] H. Okada, T. Yoshimi, M. Motokurumada, M. Ohta, K. Yamashita, *AR Navigation System Using Markers for Positioning*, in *Information Processing Society of Japan Kansai Branch Conf. Proc.* (2011)

[9] R. Yano, T. Nitta, K. Ishikawa, M. Yanagisawa, N. Togawa, *A Navigation System based on Landmark Recognition by Glass-type Wearable Devices*, in *Multimedia, Distributed, Cooperative, and Mobile Symp.* (2016), pp. 419–427

[10] *Google AI Blog: Using Global Localization to Improve Navigation*, https://ai.googleblog.com/2019/02/using-global-localization-to-improve.html, (Accessed on 12/10/2020)

[11] *Mapbox*, https://www.mapbox.com, (Accessed on 12/03/2020)

[12] R. Mannari, *Development of a navigation system that enables route guidance within the facility: Development of implementation method that realizes common operation on multiple mobile platforms*, https://www.iplab.cs.tsukuba.ac.jp/paper/reports/2014/mannari_report.pdf (2015), (Accessed on 11/03/2020)

[13] *Microsoft Soundscape — Microsoft Research*, https://www.microsoft.com/en-us/research/product/soundscape/, (Accessed on 12/10/2020)

[14] S. Son, *Research on improving GPS independent positioning accuracy using Kalman filter*, http://www.denshi.e.kaiyodai.ac.jp/jp/assets/files/pdf/investigation/20070823.pdf (2007), (Accessed on 11/05/2020)

[15] R. Ogawara, H. Hatano, M. Fujii, A. Itoh, Y. Watanabe, *Location Estimation System Based on GPS Positioning and Sensor Information*, Information Processing Society of Japan Conf. Proc. **56**, 2 (2015)