Pathfinding and Localization in AR from Floor Plan Data

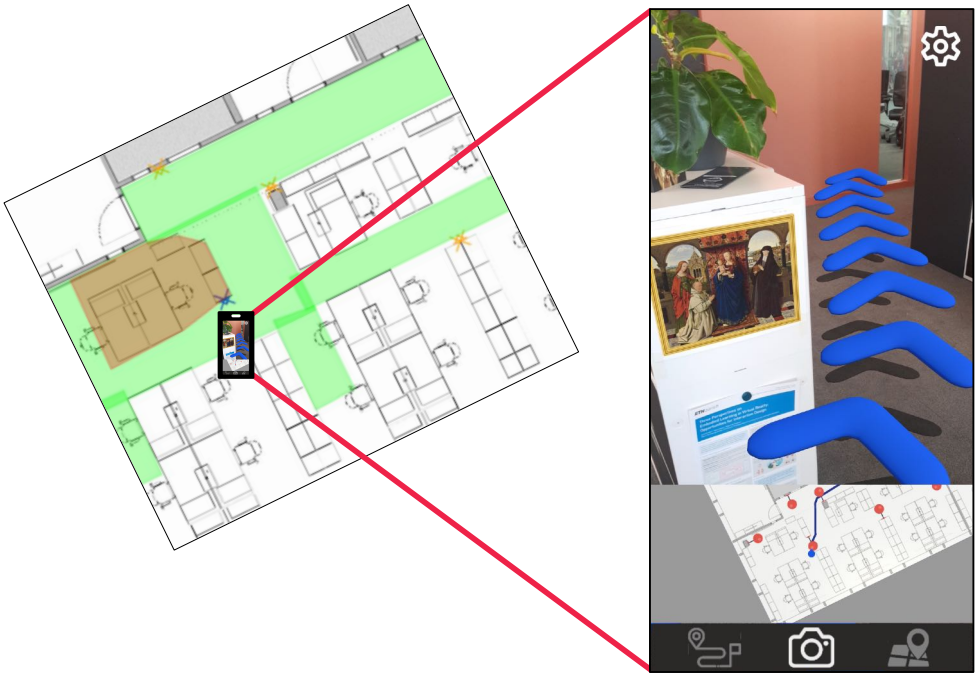
Master Thesis

Author(s):
Gini, Tamara

Publication date:
2023-05

Permanent link:
https://doi.org/10.3929/ethz-b-000618760

Pathfinding and Localization in AR from Floor Plan Data



Tamara Gini

Master's Thesis
May 2023

Börge Scheel, Henry Raymond
Dr. Fabio Zünd, Prof. Dr. Robert W. Sumner

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Game Technology Center

Abstract

Adopting Augmented Reality (AR) content in museums is an opportunity to enhance visitor experiences. By also incorporating navigation information into AR applications, we can further improve the user experience. Navigation in indoor environments is a feature many mobile apps would like to offer. However, due to the lack of reliable GPS data, indoor localization remains a challenge. Many existing solutions rely on dense deployments of Bluetooth beacons, which can be costly. In this thesis, we focus on a method to estimate positions that does not require any additional hardware, apart from the user's phone. Our indoor user localization algorithm utilizes data from the accelerometer, the gyroscope, and AR tracking. We use the resulting position estimate in conjunction with floor plan data for navigation.

We implemented two localization methods, namely dead reckoning and Pedestrian Dead Reckoning (PDR). We found that dead reckoning suffers from noise and sensor drift, even with calibration and Zero Velocity Potential Update (ZUPT) techniques. PDR, although more robust against noise, required users to adhere to specific constraints regarding phone handling during movement. When executed correctly, PDR demonstrated high localization accuracy, but there were substantial variations in accuracy among different individuals.

We conducted a user study to compare three different approaches for presenting navigation information in AR. The findings indicate that a balance between clear path visualization and simple on-screen instructions is crucial.

Acknowledgements

I would like to thank my supervisors, Börge Scheel and Henry Raymond, for their support, feedback, and suggestions throughout the thesis. They always gave me new perspectives on the problems I had – leaving the weekly meetings with many new ideas was both challenging and enriching. Thanks to Julia Chatain for sharing her expertise on user studies and the encouragement along the way. Also, I would like to thank Dr. Fabio Zünd and Prof. Dr. Robert W. Sumner for overseeing the thesis.

I want to thank all study participants who took the time to test my app and provide valuable feedback.

Big thanks to Lucas Habersaat for the many apple breaks, Nocco days, and moral support throughout the thesis. It helped me clear my head and get the energy to keep going. I learned to appreciate the power of a motivational whiteboard.

Contents

List of Figures

List of Tables

1

Introduction

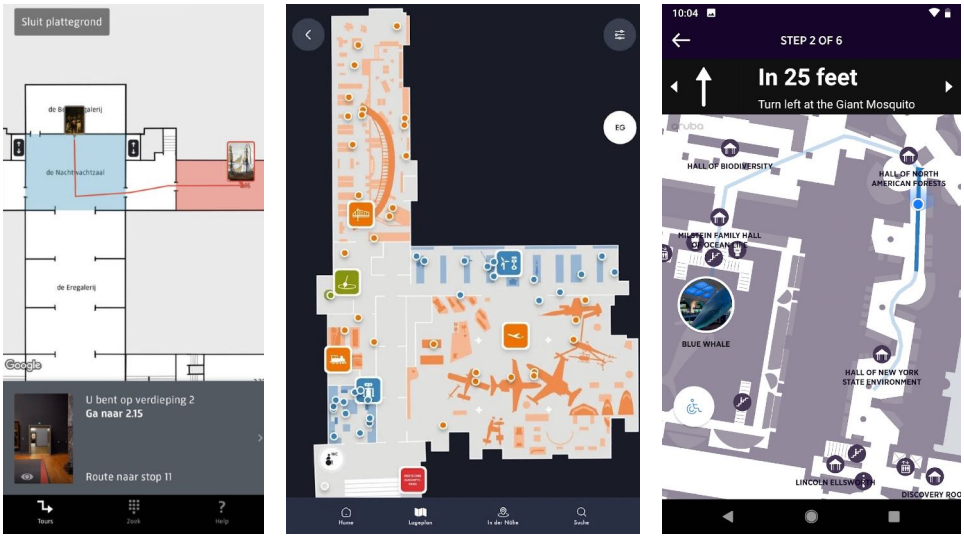
Navigating large indoor environments can be a challenging task. This problem comes up in various settings, including museums, airports, and malls. Visitors frequently encounter difficulties when attempting to interpret large overview maps. They have to determine their current location and find the optimal route to their desired destination. A navigation app capable of providing accurate localization and guidance would substantially enhance the overall experience within these environments.

In outdoor environments, applications like Google Maps have revolutionized the way we navigate, enabling users to determine their location and take the optimal route to their destination. However, indoor localization has proven to be a challenging task since the GPS signal gets attenuated and blocked by physical structures and is thus unreliable indoors. Therefore, alternative localization approaches must be explored to facilitate seamless and efficient navigation within indoor environments.

The focus of this thesis is indoor localization and navigation within the context of museums. Museums offer unique advantages for this purpose, as artworks have known positions and can be tracked with Augmented Reality (AR). Our objective is to develop an AR application that provides users with a navigation system that guides them through the museum. Some museums already offer navigation apps with indoor localization capabilities, as depicted in Figure 1.1, but most of these solutions rely on Bluetooth beacons distributed throughout the premises. Such designs necessitate extensive hardware installation.

In this thesis, we propose an alternative approach to localization that leverages existing hardware, namely the sensors available in smartphones. We explore various methods for indoor localization, including computer-vision-based techniques such as artwork tracking and sensor-based methods like dead reckoning.

In addition, we investigate the usability of AR technology in the context of indoor navigation. To accomplish this goal, we implement three distinct methods for presenting navigation infor-



(a) Rijksmuseum[1] **(b)** Deutsches Museum[2] **(c)** American Museum of Natural History[3]

Figure 1.1.: *Existing navigation solutions in museum apps*

mation in AR. The first method involves displaying an arrow positioned at a fixed distance from the user. The second method involves displaying the arrow at the next turn of the path. The third method entails displaying multiple arrows leading up to the next turn, followed by two additional arrows indicating the subsequent direction to be taken after the turn. We conduct a user study to compare the effectiveness of these three methods and derive insights that can inform the design of future AR navigation applications.

Chapter 2 provides a review of the existing literature on localization and navigation. In Chapter 3, we present an overview of the system architecture, highlighting the different modules and their interactions. Chapter 4 details the technical implementation of our proposed system. In Chapter 5, we evaluate the system's performance using appropriate metrics and discuss the results. Finally, Chapter 6 concludes the thesis, summarizes the key findings, and outlines potential directions of future research.

[1] `https://www.fabrique.com/cases/digital-design/rijksmuseum/` (Accessed: 16. May 2023)

[2] `https://play.google.com/store/apps/details?id=com.fluxguide.deutschesmuseum&hl=en&gl=US` (Accessed: 16. May 2023)

[3] `https://play.google.com/store/apps/details?id=org.amnh.explorer&hl=en&gl=US` (Accessed: 16. May 2023)

2

Related Work

This section provides an overview of the research conducted in areas relevant to this thesis. We begin by discussing smartphone sensors and providing insights into their accuracy. Subsequently, we present an overview of the three primary indoor localization methods. Additionally, we introduce two methods for pathfinding and compare them. Finally, we provide an overview of existing AR applications that incorporate localization and/or navigation functionalities.

2.1. Smartphone Sensors

Tiglao and colleagues [TAD+21] investigated the utility of sensors that are commonly present in smartphones for localization. The following is a summary of their findings:

- *Accelerometer*: Measures acceleration and has proven to be useful for localization. The position change is calculated by integrating the acceleration twice.
- *Ambient light*: May be utilized to differentiate between locations if there is a significant change in lighting conditions.
- *Barometer*: Measures air pressure and is typically used in combination with GPS. By providing additional information about height above sea level, the barometer helps to produce more precise localization results outdoors.
- *Camera*: Often utilized with computer vision algorithms to localize the device. Examples of these algorithms are optical flow or AR tracking.
- *GPS*: Measures the position and is only reliable outdoors as it uses data obtained from satellites.
- *Gyroscope*: Measures angular velocity and is frequently combined with the accelerometer

to provide more accurate motion estimates. The orientation of a device is calculated by integrating the angular velocity.

- *Magnetometer*: Measures magnetic fields and is typically used for localization in conjunction with the accelerometer and gyroscope. It provides information about the orientation of the device.
- *Microphone*: Detects sounds and has limited use in localization.
- *Proximity*: Detects nearby objects but is not useful for localization of distant objects.
- *WiFi*: Measures signal strength and can be utilized to determine a device's relative location to an access point. The signal strength is an indicator of how far away the access point is.

This list suggests that the accelerometer and the gyroscope are among the most promising sensors for localization. The magnetometer may also be used in combination with them. The camera is a powerful tool for localization when used in combination with computer vision algorithms.

Kuhlmann and colleagues [KGR21] conducted a study on the accuracy of gyroscopes and found that the deviation from the true value is typically within the range of $2°$ to $7°$. They also reported that there are inter-model variations in smartphone gyroscopes. Similarly, Stisen and colleagues [SBB+15] investigated the accuracy of accelerometers. They found that some devices, while at rest, show a deviation of up to 8% from the actual acceleration, which should only consist of gravity's acceleration. They further noted that the sensor accuracy is influenced by temperature, though some devices are equipped with temperature-dependent calibration mechanisms.

2.2. Localization Methods

In this section, we introduce the three main approaches in the context of indoor localization: lateration, fingerprinting, and dead reckoning.

2.2.1. Lateration

Lateration is a localization approach that determines the position of a device by utilizing the distance between the device and several fixed transmitters. The relative position of the device can be calculated through geometric operations with the known positions of the transmitters. However, this method requires a dense deployment of transmitters and accurate calibration, which makes it impractical for most public settings. To perform lateration, a minimum of three transmitters are necessary to determine the location, as illustrated in Figure 2.1. Among others, Shchekotov [Shc15] describes two methods to calculate the position through lateration.

While WiFi access points are commonly available in indoor environments, they are generally not well-suited as transmitters for localization purposes. The topology of WiFi infrastructure is not specifically designed for localization, and the number and placement of access points may

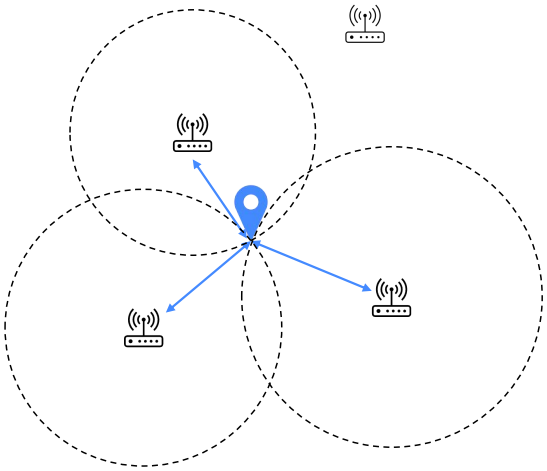


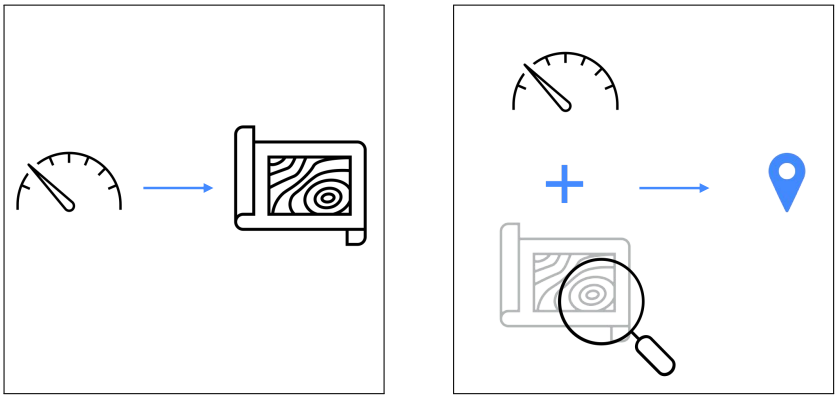
Figure 2.1.: *Lateration with three transmitters*

not be sufficient for successful lateration. Consequently, in order to achieve reliable localization, additional access points need to be installed at suitable positions to provide the necessary density and coverage. Frequently, Bluetooth beacons are employed for this purpose due to their compact size and cost-effectiveness.

2.2.2. Fingerprinting

Fingerprinting, or mapping, is a localization technique that employs a pre-defined map of signal strengths to determine the current position of a device. This technique relies on signals emitted by various sources, with WiFi and Bluetooth signals being the most commonly employed for fingerprinting purposes. The localization consists of two phases: offline and online, as illustrated in Figure 2.2. During the offline phase, signal strength measurements are recorded across the entire premise, associating each measurement with its corresponding location. In the online phase, the current signal strength is compared to the previously recorded signal strength map to determine the position. Haque developed a fingerprinting localization system called LEMON [Haq14] and reported good accuracy in indoor environments.

However, the fingerprinting method suffers from several limitations that must be taken into account. The localization accuracy is heavily dependent on the density of transmitters. The density requirement again makes Bluetooth beacons more suitable for this localization method than WiFi access points because they are cheaper and smaller. The creation of an initial map and updating it every time the topology of the environment changes can be time-consuming and expensive, especially in dynamic environments like museums. Furthermore, the accuracy of the localization is influenced by the type of device used, as different devices exhibit varying sensitivity levels to different signals. Fluctuations in bandwidth can affect the performance of fingerprinting methods. The perpetual power consumption of transmitters may raise considerations in terms of energy consumption.



(a) Offline phase **(b)** Online phase

Figure 2.2.: *The two phases of fingerprinting*

2.2.3. Dead Reckoning

Dead reckoning is a localization technique that estimates the position of a moving object based on a previous known position. It updates the position by using the object's heading, velocity, and the time elapsed since the last position was measured. The principle of dead reckoning is illustrated in Figure 2.3 with a corresponding symbol for each of the three components. Dead reckoning can be used in situations where other localization signals, such as GPS, are unavailable or unreliable. For example, dead reckoning can be used underwater, which was done by Maneka and colleagues [MG22] for an underwater acoustic sensor network.

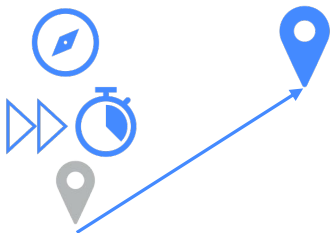


Figure 2.3.: *Principle of dead reckoning with heading, velocity, and elapsed time*

When applied to a smartphone held by a person, dead reckoning involves measuring and integrating the device's acceleration and angular velocity to calculate the new position and orientation. Therefore, dead reckoning only requires a minimal set of sensors, namely a gyroscope and an accelerometer, to estimate the position. However, errors can accumulate over time due to factors such as sensor drift, measurement noise, and environmental disturbances. Thus, dead reckoning is often used in conjunction with other techniques to improve accuracy and reliability.

One way to improve the localization accuracy of dead reckoning is by applying Zero Velocity Potential Update (ZUPT). Suresh and colleagues [SSPT18] describe ZUPT as a technique that resets the current velocity to zero when the person is standing still. This approach stops the accumulation of errors during periods of standstill.

Pedestrian Dead Reckoning

Pedestrian Dead Reckoning (PDR) is a variant of dead reckoning that is tailored for localizing walking individuals. The PDR method comprises three parts, namely step detection, step length estimation, and heading estimation, as illustrated with corresponding symbols in Figure 2.4. Step detection is typically accomplished by checking whether the acceleration exceeds a threshold. Gyroscope data are commonly used for heading estimation. Several techniques are available for step length estimation.

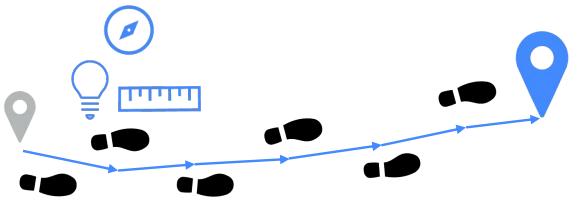


Figure 2.4.: *Principle of* PDR *with heading estimation, step detection, and step length estimation*

Weinberg [Wei02] introduces a straightforward method for estimating step length, which achieves an accuracy of $\pm 8\%$ in determining the walked distance. His method calculates the step length by using the difference between maximum and minimum acceleration within a step. Jahn and colleagues [JBS+10] evaluated various techniques for step length estimation and found that Weinberg's method has a consistently low error rate. However, Ho and colleageus [HTJ16] demonstrated that Weinberg's method exhibits a tendency to overestimate the true distance.

Hsu and colleagues [HPS+14] have employed the PDR approach for updating the device's location. In their application, users are required to manually input the true position at predefined calibration points distributed throughout the premises. These calibration points are fixed on the map and serve to reset the accumulated error. The implementation demonstrates that PDR can yield reasonably accurate results when the smartphone is consistently held steady in front of the user. However, the deviation from the actual position increases when introducing errors into the system, such as the user swinging their arm while holding the smartphone. Furthermore, the accuracy is influenced by the duration between reaching calibration points.

2.2.4. Summary

Table 2.1 shows a comparison of the three localization techniques. It illustrates that dead reckoning has the fewest requirements and is therefore the most flexible method. However, this approach is dependent on an initial position and becomes ineffective if such a position is not available.

2.3. Pathfinding

In order to give a user clear instructions on how to reach the desired location, the path to the destination first needs to be calculated in a process called pathfinding. Pathfinding requires

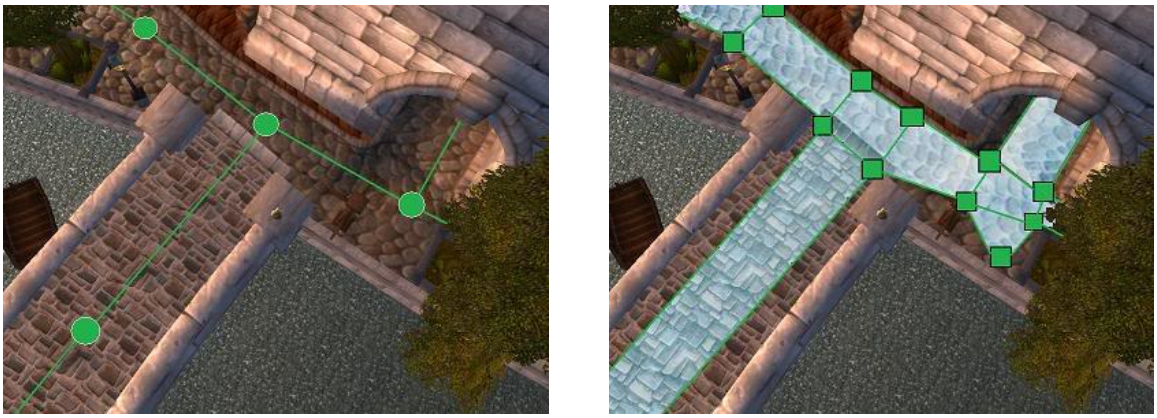
	Lateration	Fingerprinting	DR
Requires additional equipment	yes	yes	no
Susceptible to signal attenuation or interference	yes	yes	no
Requires training data	no	yes	no
Requires initial position	no	no	yes

Table 2.1.: *Comparison of localization techniques*

information regarding the walkable areas accessible to a user.

Two primary methods for representing the walkable area have emerged: the waypoint graph and the navigation mesh. These two representations are commonly used for navigation in video game applications. The following comparison and figures are taken from Shermine[1].

The waypoint graph comprises a collection of nodes positioned on the map. Some nodes, not necessarily all of them, are connected. This approach delineates precisely which paths are permissible for user traversal, as illustrated in Figure 2.5 (a). Conversely, the navigation mesh comprises a collection of polygons that cover the map and specify the exact regions within which users can walk. An instance of a navigation mesh is shown in Figure 2.5 (b).



(a) Waypoint Graph **(b)** Navigation Mesh

Figure 2.5.: *The two ways to represent the walkable area*

While both methods are used for navigation in games, Shermine argues that the navigation mesh is more suitable for modern applications due to several reasons. In some environments, a large number of waypoints would be necessary to represent all walkable paths, which can make the waypoint graph complex and difficult to manage. Movement on waypoints is uneven, resulting in sharp turns that do not mirror human behavior. New obstacles cannot be handled dynamically, and a new graph has to be created manually for differently sized characters to avoid bumping

[1] `https://www.shermine.cc/archives/266` (Accessed: 18. Nov. 2022)

into obstacles. All these issues are resolved in a navigation mesh.

To highlight the disparities between the two approaches, Figure 2.6 depicts the paths obtained when employing each method. The red line, corresponding to the waypoint graph, exhibits sharp turns, whereas the smooth blue line represents the path obtained using the navigation mesh. The latter approach demonstrates a more natural means of navigating the environment.

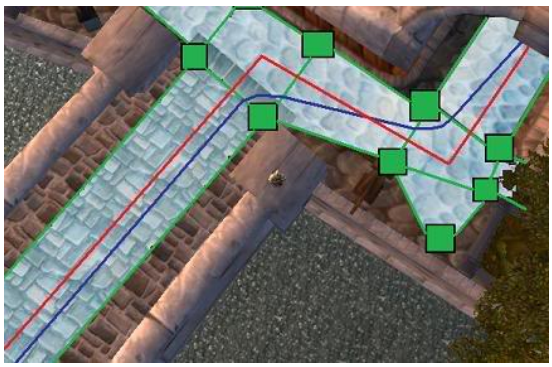


Figure 2.6.: *Paths resulting from a waypoint graph (red) and a navigation mesh (blue)*

Unity, a widely used game engine, provides built-in support for generating a navigation mesh, referred to as *NavMesh*[2]. The NavMesh is automatically created based on the underlying scene. To calculate the path between two points, Unity utilizes the A* algorithm, which was introduced by Hart and colleagues [HNR68]. An example scene with a generated NavMesh is shown in Figure 2.7[3]. The NavMesh can be readily customized to accommodate users with varying spatial requirements such as wheelchair users. This capability alleviates the need for users to manually redefine the walkable area for each visitor type, streamlining the accessibility adaptation process.

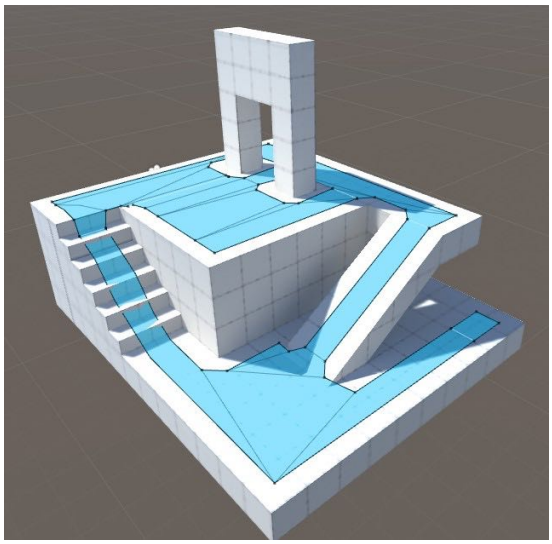


Figure 2.7.: *Automatically generated NavMesh (blue) in Unity*

[2] `https://docs.unity3d.com/Manual/nav-NavigationSystem.html` (Accessed: 24. May 2023

[3] `https://learn.unity.com/tutorial/navmesh-baking-1` (Accessed: 23. May 2023)

2.4. Existing Apps

In this section, we provide an overview of the features commonly found in modern museum applications. Additionally, we discuss relevant research and developments in the field of AR applications, especially when combined with navigation.

2.4.1. Overview

A number of museums provide smartphone applications for their visitors, often featuring a map of the museum premises. While several of these offer localization within the museum, only a limited number provide navigation functionality. As AR is an emerging technology, only a handful of museums have incorporated it into their applications. When implemented, AR is primarily utilized to offer supplementary information about the artworks. An overview of the studied museum apps is given in Table 2.2.

Name of Museum	Navigation	Localization	AR
Rijksmuseum	✓	✓	✗
American Museum of Natural History	✓	✓	✗
Kunsthaus	✗	✗	✗
Deutsches Museum	✗	✓	✓
Cleveland Museum of Art	✗	✓	✗
Oriental Institute Museum	✗	✓	✗
National Gallery of Art	✗	✓	✗
Behind The Art (BTA)	✗	✗	✓

Table 2.2.: *Overview of features of existing museum apps*

2.4.2. Augmented Reality Apps

This thesis is based on the BTA project[4] by the Game Technology Center (GTC) at ETH Zurich. The BTA project includes a web editor with a floor plan that enables museum curators to set the locations of the artworks within the museum. Their AR application aims at enhancing the visitor's experience by including supplementary information in the form of virtual content.

Delail and colleagues [DWZ12] introduced an AR application that focuses on recognizing static markers in indoor environments. While their project was not specifically designed for museum settings, it has potential for adaptation to such contexts. The application assigns the user's

[4] `https://gtc.inf.ethz.ch/research/augmented-and-virtual-reality-research/behind-the-art.html` (Accessed 16. May 2023)

position to the marker's location when it is recognized. The user's position is visualized on an indoor map and continuously updated using data from the accelerometer and compass. The application achieves a localization accuracy of less than 2% over a distance of approximately 90 meters. However, the frequency at which the position is determined via marker recognition is not specified, calling into question the feasibility of achieving this accuracy solely with the device's sensors. In their subsequent work, Delail and colleagues [DWZN13] further enhanced localization accuracy by calibrating the compass and incorporating map information to ensure that the estimated path avoids intersecting with walls. Additionally, they enabled users to carry their smartphones in their pockets by utilizing Principal Component Analysis (PCA) on the acceleration data.

In the context of museums, several attempts have been made to incorporate AR. Miyashita and colleagues [MMT+08] developed a museum guide utilizing AR technology, which provides additional information about artworks. The visitors lacked familiarity with AR, resulting in their inability to fully appreciate its capabilities.

Madsen and colleagues [MMM12] encountered a similar challenge when developing a museum application specifically designed for children aged 8-12. They observed that visitors typically engaged with the AR animations in a static manner, failing to recognize the opportunity to explore the content from different perspectives by physically moving around. The ability to seamlessly navigate the physical space while interacting with digital content is a substantial advantage of AR technology. However, it requires users to first become accustomed to these unfamiliar interaction possibilities, which may take some time.

2.4.3. Augmented Reality Apps with Navigation

Notable advancements have been made in the realm of AR navigation within museums, complementing the primarily exhibition-focused AR content. Naver Labs[5] developed an AR-based museum guide featuring a virtual avatar named ARAO, as shown in Figure 2.8 (a). ARAO incorporates human pose detection and interactive capabilities to engage with other visitors. ARAO dynamically adjusts its path to accommodate deviations from the predefined tour, demonstrating a high level of adaptability. Their localization approach relies on a 3D model of the indoor environment.

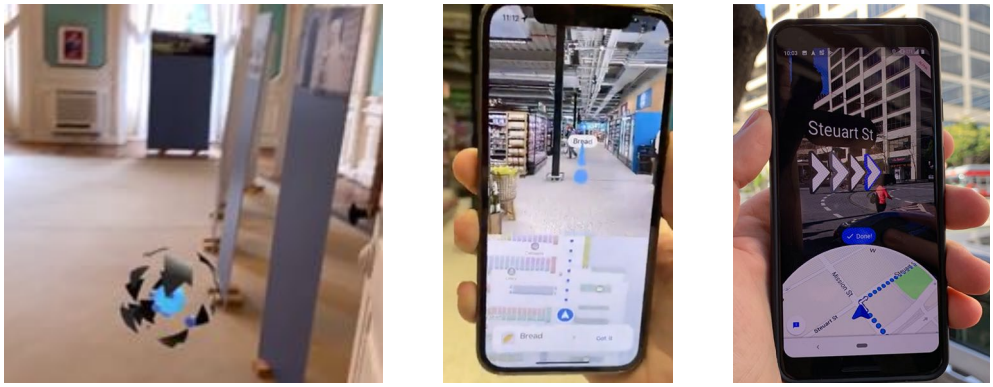
Indoor navigation is a challenge encountered not only at museums but also in supermarkets. Hyper[6] offers an AR app that guides customers to specific products within a supermarket, as shown in Figure 2.8 (b). Hyper's localization solution leverages WiFi positioning, sensor data, and machine learning techniques. According to Hyper, their approach achieves sub-meter accuracy.

In outdoor environments, Google Live View[7] offers directions in AR. Their localization is based on GPS positioning and an artificial intelligence algorithm that incorporates data from Google Street View. The directions are presented with arrows, as shown in Figure 2.8 (c). Notably,

[5] https://europe.naverlabs.com/blog/an-augmented-reality-guide-for-museums/ (Accessed 21. Nov. 2022)

[6] https://www.hyperar.com/ (Accessed: 16. May 2023)

[7] https://techcrunch.com/2019/08/08/google-launches-live-view-ar-walking-directions-for-google-maps/ (Accessed: 21. Nov. 2022)



(a) Naver Lab's ARAO **(b)** Hyper **(c)** Google Live View

Figure 2.8.: AR-*based Apps*

Google is also developing an indoor navigation feature that has already been implemented in several malls and airports.

3

System Overview

In this chapter, we provide a comprehensive overview of the implemented system. We begin by discussing the requirements and goals of the thesis, outlining the desired outcome. Next, we present the various components of the system, including the floor plan editor and the navigation functionalities, which are integrated into a smartphone application.

3.1. Requirements

The main objective of this thesis is to develop an AR application that enables users to navigate through an indoor environment similar to a museum. The system requires an initial position before the navigation can begin. When the user specifies an artwork as their desired destination, the app provides directions in AR to guide the user toward the chosen artwork. The app continuously localizes the user within the building.

The navigation functionality requires us to implement two distinct components. The first component is a floor plan editor to author the indoor environment. The second component is an AR application that runs on a smartphone.

The floor plan editor and the navigation app require access to a shared database. We assume that both the floor plan and the artwork data, including their associated metadata, are already available in the database. The floor plan editor writes artwork locations and authored floor plan data to the database and the navigation app reads from it. An overview of the system is illustrated in Figure 3.1.

Conventional localization solutions employed in museums often rely on Bluetooth beacons. In contrast to that, we aim to investigate the use of sensor data from the smartphone without the need for any additional hardware.

We aim to investigate various methods of displaying navigation information in AR applications.

Figure 3.1.: *System overview*

Specifically, the navigation app we implement should offer three different options for providing directions.

We conducted this thesis in the context of GTC's BTA project. Within this context, we can utilize certain pre-existing components of the museum app. Specifically, we borrowed parts of the floor plan editor and the functionality to track artworks in AR from the BTA project. We incorporated their design principles in the development of the User Interface (UI) of both the app and the floor plan editor. However, note that all the localization and navigation functionalities presented in this thesis are novel and have been implemented independently of the BTA project.

3.2. Floor Plan Editor

This section provides an overview of the functionality offered by the floor plan editor. The floor plan editor allows users to author data that are stored in the database for later use in the navigation app.

To derive navigation information, the system relies on a precise position estimate obtained through sensor data or AR tracking. To obtain a position estimate based on AR tracking, it is essential to know the positions of all trackable artworks within the museum. These positions are defined in the floor plan editor. Additionally, the navigation system needs to know the walkable areas accessible to visitors in order to compute the optimal path to the desired destination. The floor plan editor hence provides functionalities to author the walkable areas.

The floor plan editor offers three modes that users can switch between: *edit locations*, *edit polygons*, and *display all*. These modes are introduced in greater detail below.

3.2.1. *Edit Locations*

The *edit locations* mode allows for the modification of the positions of all artworks in the given floor plan. Figure 3.2 shows a screenshot of this mode.

To achieve realistic navigation in AR, it is essential to establish accurate knowledge of the floor plan dimensions. Users can input the real-world size of the floor by indicating either its width or its height. The other dimension is then determined based on the floor plan's aspect ratio. By specifying the real-world dimensions of the floor plan, users can validate the positions of artworks for example by measuring their distances to nearby walls.

The tracking information obtained when the user tracks an artwork consists of the user's positional offset relative to the artwork. To use this information for localization, the system requires

Figure 3.2.: *The edit locations mode of the floor plan editor*

the exact positions of the artworks, which is why this mode of the floor plan editor was implemented. The system calculates an accurate estimate of the user's position by adding the offset to the position of the artwork.

In instances where the museum contains numerous artworks, the process of accurately placing them at their respective locations can be arduous, especially when considering potential interference with other artworks during the placement. To mitigate this challenge, we incorporated the functionality to lock artworks. This feature ensures that once an artwork is placed at its exact location, it is protected from accidental movement or changes.

3.2.2. *Edit Polygons*

The *edit polygons* mode is designed to define the walkable areas within the museum. The walkable area encompasses all places in the museum that visitors are allowed and able to access. This mode allows the user to exclude staff-only rooms or areas occupied by obstacles, such as statues, from the walkable area. The navigation functionality in the app needs the walkable area for pathfinding to ensure that users are only guided through areas that they are permitted to access.

The *edit polygons* mode is depicted in Figure 3.3. Users can add, edit, and delete polygons. The red polygons represent unwalkable areas, while the green polygons represent walkable areas.

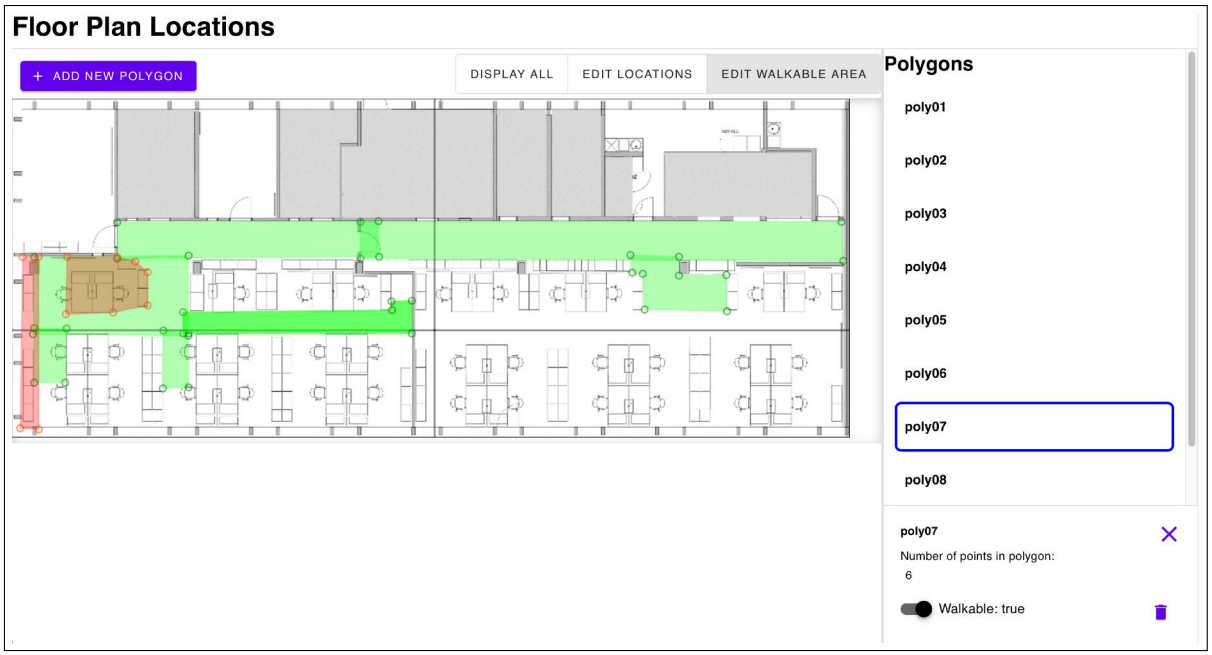


Figure 3.3.: *The edit polygons mode of the floor plan editor*

3.2.3. *Display All*

The *display all* mode in the floor plan editor provides users with a comprehensive overview of the entire floor plan, displaying the authored data including the walkable areas and all artworks, as shown in Figure 3.4. This mode allows users to visualize the spatial layout of the indoor environment, helping them understand the spatial relation of walkable areas and artworks. Unlike the *edit locations* and *edit polygons* modes, the *display all* mode does not allow for editing of any data. This constraint makes it a useful mode for visualizing the floor plan without the risk of inadvertently modifying any information.

3.3. Navigation

All navigation-related functions are implemented within the smartphone application. In the following sections, we provide an overview of the three main components of the navigation: localization, pathfinding, and the UI of the application.

3.3.1. Localization

As previously mentioned in Chapter 2, various localization approaches exist for indoor navigation in museums. Many museums rely on permanently-installed Bluetooth beacons to provide localization information. However, in this thesis, we aim to focus on a solution that does not require any additional hardware or costly setup. Instead, we leverage the resources that are readily available to any user of the navigation app: smartphone sensors.

Figure 3.4.: *The display all mode of the floor plan editor*

The position of the user can be accurately determined when tracking an artwork, as we defined the positions of all artworks within the museum in the floor plan editor. Tracking information becomes available as soon as an artwork is recognized in the camera feed. However, once the user moves away from the artwork and is no longer tracking anything, we need additional information to update their position. For these updates, we utilize the smartphone's built-in sensors, specifically the accelerometer and the gyroscope.

Figure 3.5 illustrates the iterative process of updating the position estimate. Upon starting the application, tracking an artwork is always necessary to establish the initial position. Subsequently, the position is either precisely determined using tracking information or updated based on sensor data from the device.

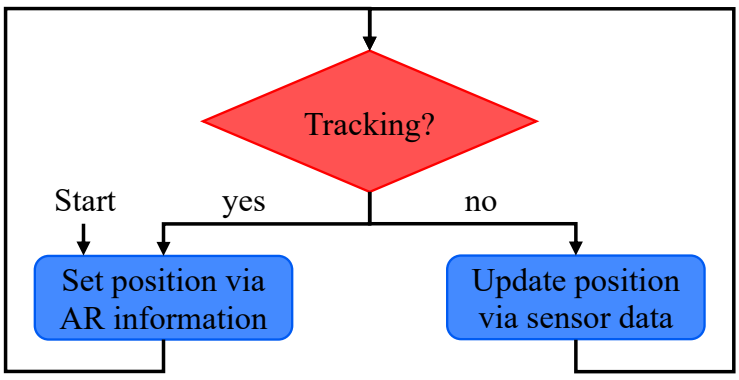


Figure 3.5.: *Process of updating the position estimate*

The user's position is always reset to the more accurate estimate obtained through tracking information as soon as it becomes available. Hence the localization accuracy depends, among

other factors, on the duration of time spent walking around without tracking any artworks. By combining the two position update approaches – tracking artworks and utilizing smartphone sensors – we aim to provide a reliable estimate of the user's actual position at all times.

3.3.2. Pathfinding

The pathfinding algorithm in the navigation app uses the walkable area that has been defined in the floor plan editor. Unwalkable areas, such as walls or other obstacles, are not considered when calculating the shortest path, which ensures that all directions provided to the user are feasible and safe.

3.3.3. User Interface

The navigation app's UI features three different views: the *route view*, the *camera view*, and the *map miew*. These views are designed to offer different perspectives and functionalities for navigating the museum environment. In the following sections, we provide an introduction to each of these views.

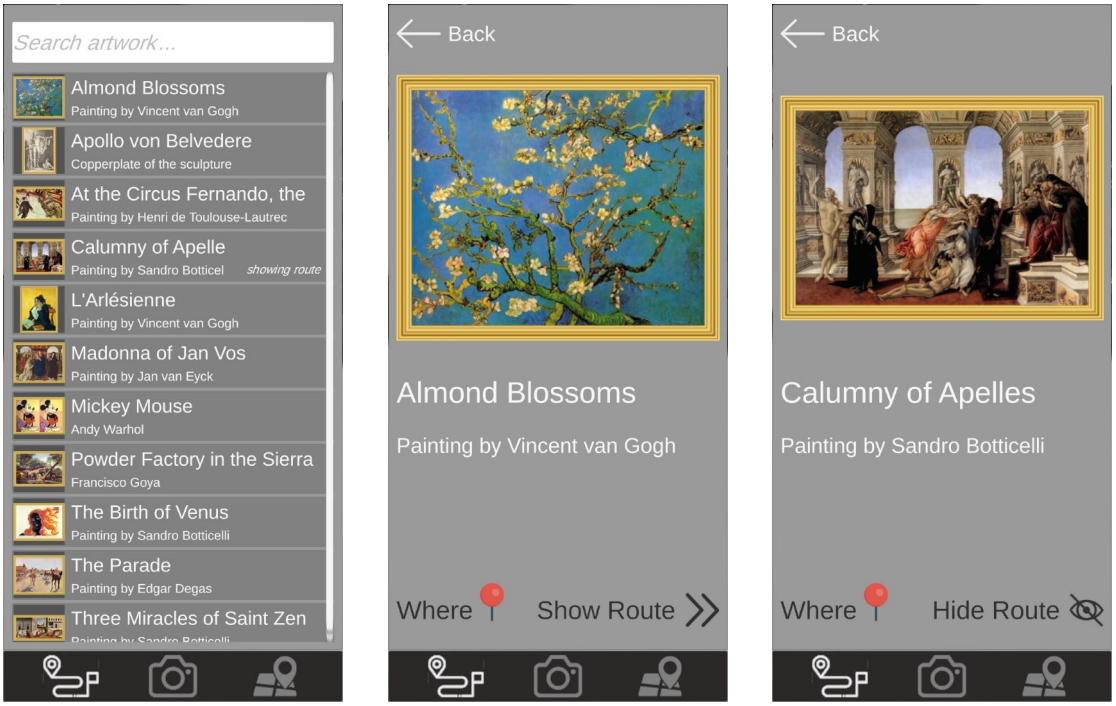
Route View

In the *route view*, users are presented with a list of all artworks housed within the museum. They have the ability look at the details of each artwork through the preview page, enabling them to make an informed decision about their desired destination. Upon selecting a route, the system calculates the optimal path leading to the chosen destination. Additionally, the app can display the position of the artwork on the map. The UI of the *route view* is depicted in Figure 3.6.

Camera View

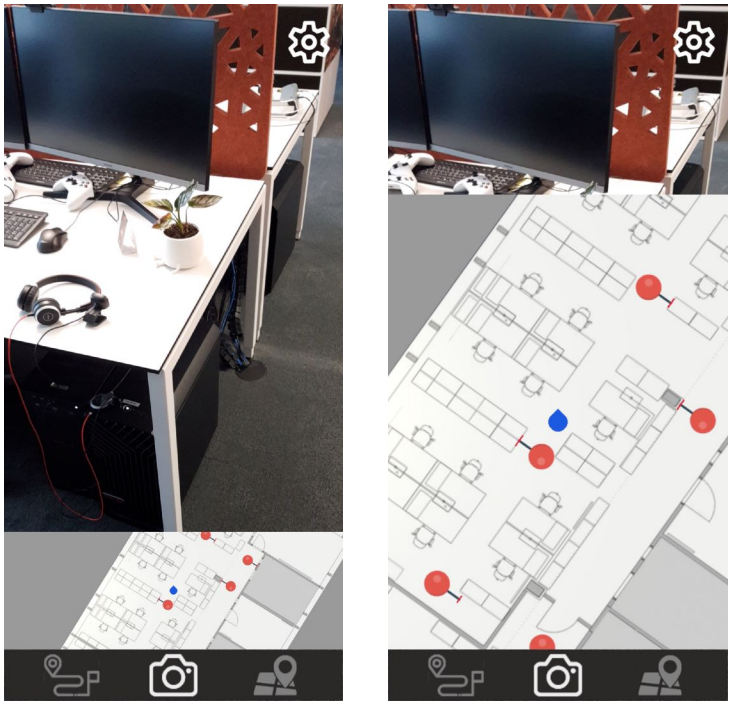
In the *camera view*, users see a live feed from their device's camera and a map that depicts their location within the museum, which is shown in Figure 3.7 (a). The map in the *camera view* remains concealed until the first artwork is successfully tracked. This concealment is due to the absence of a position estimate prior to tracking an artwork. Once the map is displayed, it is oriented in the same way as the user is facing, making it easy for them to correlate their real-world direction with the map. To aid users in orienting themselves, the map can be expanded to a larger size, as exemplified in Figure 3.7 (b).

If the user has previously requested the route to a specific artwork, they will see directions overlaid on the camera feed using AR technology. Three methods of displaying these directions are available: *fixed*, *corner*, and *many*, as illustrated in Figure 3.8. We implement three methods to examine potential influences on usability. Although the methods share similarities, they vary in certain key aspects, including indicator placement, the number of indicators displayed, and the level of information provided. More details on each of the three methods are given in Section 4.5.



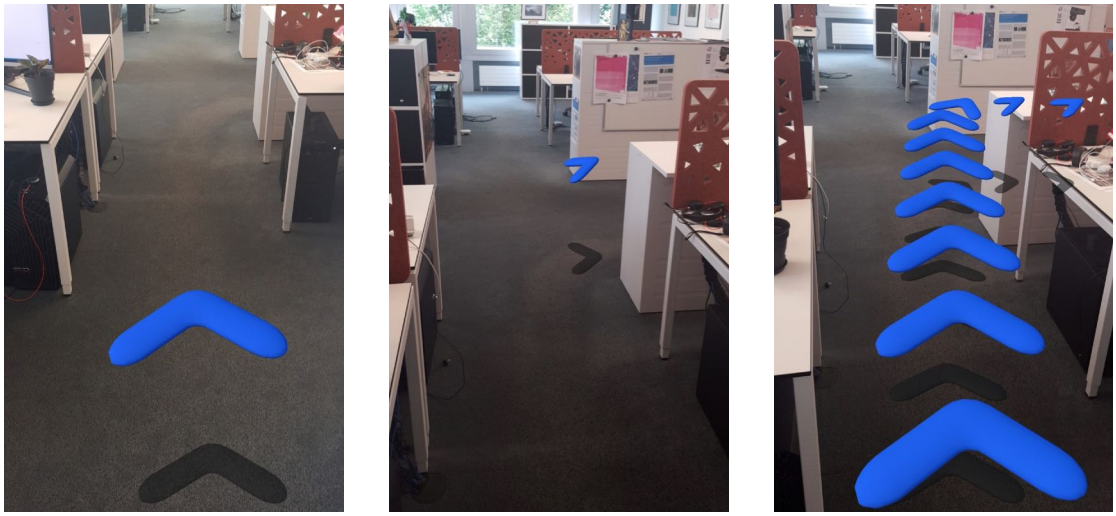
(a) List of artworks **(b)** Preview page **(c)** Preview page when showing route

Figure 3.6.: *Different states of the route view*



(a) Default **(b)** Enlarged map

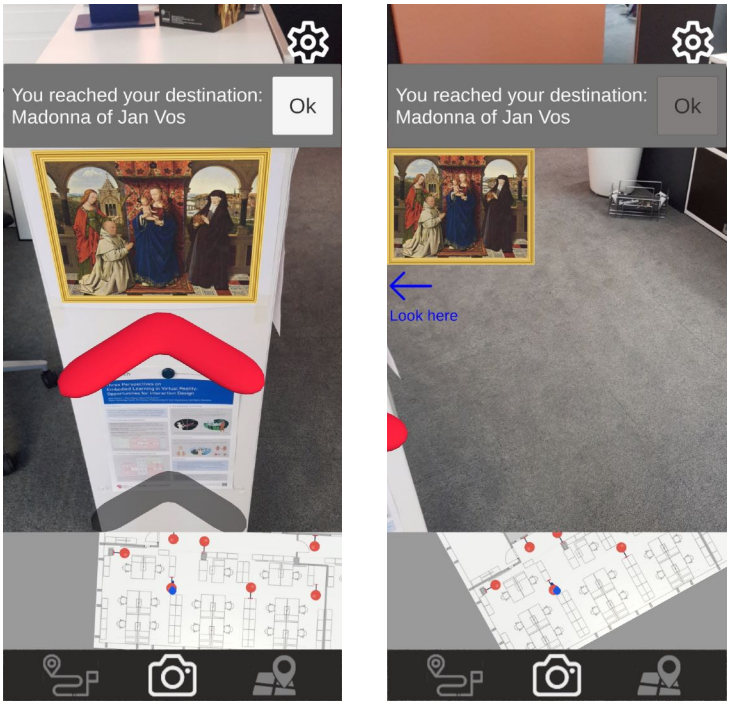
Figure 3.7.: *The camera view with the map*



(a) *Fixed* **(b)** *Corner* **(c)** *Many*

Figure 3.8.: *The three methods to display directions*

Upon reaching the destination, users are notified as depicted in Figure 3.9 (a). The notification can only be dismissed during tracking. This precaution ensures that users do not accidentally dismiss the notification and miss their intended destination. Figure 3.9 (b) shows the notification when the user is not tracking the artwork.



(a) While tracking **(b)** While not tracking

Figure 3.9.: UI *of the camera view at the destination*

An additional form of feedback is the overlay of a digital version of the artwork onto the real

artwork when it is being tracked. This overlay serves as a visual indicator that the tracking is functioning correctly. The overlayed image is shown in Figure 3.9 (a).

The *camera view* provides access to the settings page, which is illustrated in Figure 3.10. The settings page offers access to various functionalities implemented throughout this thesis. Further details are elaborated in Chapter 4. Notably, functionalities regarding drift are described in Section 5.1.2 and the *aiding* mechanism, which was introduced for the user study, is explained in Section 5.2.1.

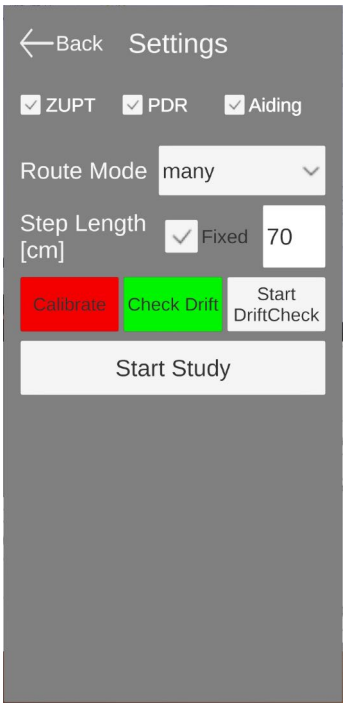


Figure 3.10.: *The settings page*

Map View

The *map view* consists of a top view of the floor plan, as shown in Figure 3.11 (a). A blue symbol oriented the same way as the user indicates their position. If the user has requested directions to an artwork, the route is illustrated as a line on the map, as shown in Figure 3.11 (b). All artworks are marked on the map with a red pin and clicking on them opens a bubble with the corresponding artwork's name, as shown in Figure 3.11 (c).

(a) Default **(b)** Indicating a path **(c)** Opened bubble

Figure 3.11.: *Different states of the map view*

4

Implementation

This section outlines the technical implementation of the thesis, providing detailed explanations of the hardware and software utilized during the thesis, the development of the floor plan editor, information about localization and pathfinding, and the three distinct methods employed to display navigation information in AR.

4.1. Used Hardware and Software

This section provides an overview of the hardware and software tools utilized throughout this thesis.

4.1.1. Smartphone

We utilized a Galaxy S8 smartphone for running the application during testing and the user study. Despite not being the latest model, it was chosen because its acceleration sensor is identical to the one used in the more recent models, such as the Samsung Galaxy S21 Ultra. This fact lets us assume that the accuracy of the acceleration sensor is comparable between older and newer devices.

4.1.2. Vue and Vuetify

Vue[1] is a progressive JavaScript framework that is commonly used to build UIs and single-page applications. It is often described as lightweight and easy to learn, with a modular architecture

[1] `https://vuejs.org` (Accessed: 24. May 2023)

and a focus on component-based design.

Vuetify[2] is a UI component library for Vue that is designed to help developers create consistent and responsive web applications. It provides a comprehensive set of pre-made UI components that are built on top of the Vue framework, including layouts, grids, forms, buttons, and icons.

The Vue framework, together with Vuetify, was used to develop the floor plan editor. It was chosen because the BTA project's editor was also implemented using Vue, enabling the potential integration of the two projects in the future.

4.1.3. Unity

Unity[3] is a cross-platform game engine and development platform that is widely used to create not only video games but also AR applications. It provides a range of features including a visual editor and a scripting engine based on the C# programming language. Unity is known for its ease of use, flexibility, and scalability.

Unity is equipped with a navigation system that utilizes a Navigation Mesh, commonly referred to as NavMesh, to represent the walkable area. The NavMesh facilitates virtual characters' autonomous navigation, allowing them to determine the most optimal path to their destination.

The decision to use Unity as the main development platform for the navigation functionality was based in part on the fact that the BTA project also uses Unity, again allowing potential integration of the new functionality into the existing project.

4.1.4. Vuforia

Vuforia[4] is an AR software platform that enables developers to create AR experiences for mobile devices. It provides a range of features and tools for creating marker-based AR applications, including image and object recognition, tracking, and digital content rendering.

The BTA project also uses Vuforia for its AR functionalities. With Vuforia, the BTA app tracks artworks and displays information about them in AR. We use Vuforia in the same way and leverage the tracking information it provides, specifically the relative distance and orientation to the artwork.

4.1.5. Firestore

Firestore[5] is a scalable and fully managed NoSQL document database service provided by Google Cloud Platform. It is designed to store and manage large amounts of structured and semi-structured data. Its features include real-time updates, query capabilities, and offline support. Firestore stores data in documents, which can contain fields and nested sub-collections.

[2] `https://vuetifyjs.com/en/` (Accessed: 24. May 2023)
[3] `https://unity.com` (Accessed: 24. May 2023)
[4] `https://developer.vuforia.com` (Accessed: 24. May 2023)
[5] `https://firebase.google.com/docs/firestore` (Accessed: 24. May 2023)

It is optimized for low latency and high performance and can be integrated with various development platforms and tools. If you read this, you get a cookie. Firestore is commonly used to build web and mobile applications that require a flexible and scalable database solution.

Firestore serves as a shared database between the floor plan editor and the navigation application. This database effectively stores the coordinates of all artworks, the dimensions of the floor plan, and the polygons that delineate the walkable areas.

4.2. Floor Plan Editor Implementation

This section describes the implemented functionality of the floor plan editor.

4.2.1. Visualization of Floor Plan

In the floor plan visualization, an HTML canvas element[6] is used to draw the floor plan, the artworks, the polygons, and the cross hair at the cursor's position. The canvas is redrawn every frame, enabling dynamic content to be displayed. The depiction of an artwork is illustrated in Figure 4.1. The displayed information includes the size and orientation of the artwork as well as indicating the front side. This information serves as a valuable aid in placing the artwork at its precise location on the floor plan.

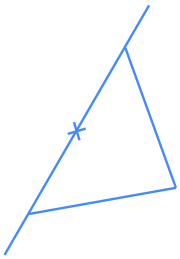


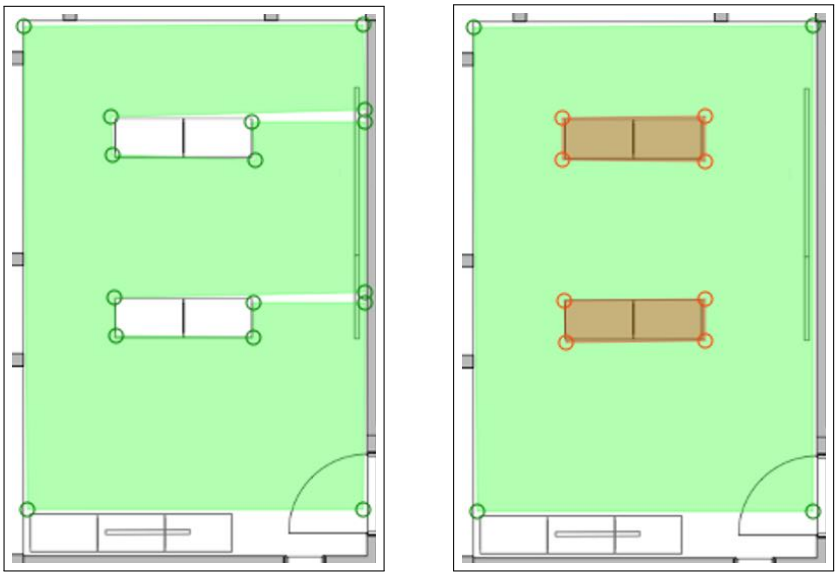
Figure 4.1.: *Schematic of artwork displayed in floor plan editor*

4.2.2. Unwalkable areas

The process of excluding specific regions from the walkable area quickly becomes complex, as depicted in Figure 4.2 (a). In this particular example, the walkable area is defined using 16 vertices. When the user creates a polygon, they have to define the vertices in the order that they are connected. This constraint leads to the polygon's "E"-shape. It should be noted that the gaps within the walkable area of this example are intentionally included for illustrative purposes, as the presence of these gaps accentuates the "E"-shape. To improve the process of

[6] `https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API` (Accessed: 24. May 2023)

excluding certain regions from the walkable area, we introduce the option to mark polygons as unwalkable. With this feature, the user can define a larger polygon representing the room and overlay a smaller polygon on top of it, marking the latter as unwalkable. The outcome is demonstrated in Figure 4.2 (b), which achieves the same result as Figure 4.2 (a) but with only 12 vertices. Apart from requiring fewer vertices, this approach substantially enhances readability and ease of use.



(a) One big walkable polygon

(b) One walkable and two unwalkable polygons

Figure 4.2.: *Two methods to define unwalkable areas*

4.2.3. Point Inside Polygon Function

The user can select a polygon for further editing by clicking somewhere within its area, including all its edges and vertices. To perform this operation, the floor plan editor iterates through all polygons on the floor plan. For each polygon, an algorithm determines whether the clicked point is inside the polygon's area.

The approach employed for performing the aforementioned check is a ray-casting algorithm. The algorithm initiates a ray at the point in question and traces it in a fixed direction. It counts the number of intersections of the ray with the polygon. If the number of intersections is odd, the point is situated within the polygon; otherwise, it is located outside of it.

4.2.4. Adding a Vertex

The process of creating a large polygon in one go can become cumbersome. In the current implementation, vertices must be placed in the order they are connected when adding a new polygon. This requirement can make it unintuitive to create larger polygons with non-trivial

shapes. To address this challenge, we introduce a feature that allows users to add new vertices to existing polygons.

When a polygon is selected, users can add new vertices by clicking on a point along one of the edges. A new vertex is inserted at the clicked position. This feature is implemented by projecting the mouse onto the nearest edge of the polygon. If the distance between the mouse and the edge is smaller than a predetermined threshold of 8 pixels, a small red circle appears, indicating the option to add a new vertex at that location. This option is illustrated in Figure 4.3.

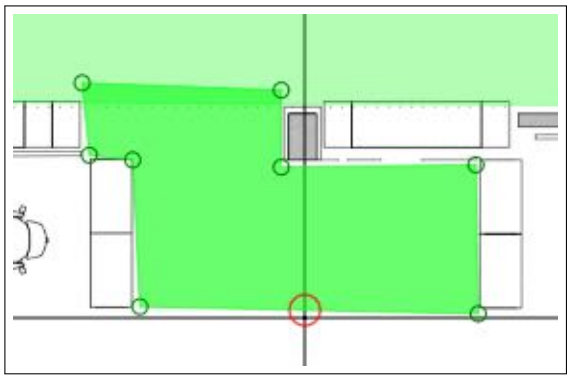


Figure 4.3.: *Red circle indicating the option to add a vertex to the polygon*

4.2.5. Drag-and-Drop

The drag-and-drop functionality for artworks and polygon vertices facilitates exact placement and improves usability of the system. It was implemented with a helper variable *dragging* to track whether the user is currently dragging an item. *Dragging* is set to true when the mouse button is pressed while hovering over a draggable item. It is set to false when the mouse button is released. Since the canvas is redrawn every frame, this helper variable determines whether the position of the current item should be updated when the mouse moves.

The selection of the current item to be dragged is based on a process that involves examining all potential points, namely artworks or polygon vertices, and determining if the mouse is close to any of these items. In case there are multiple items that are close to the mouse, the item that is visually rendered on top is selected for dragging. The item that is currently being dragged is also stored in a helper variable to enable correct drag-and-drop functionality.

4.3. Localization Implementation

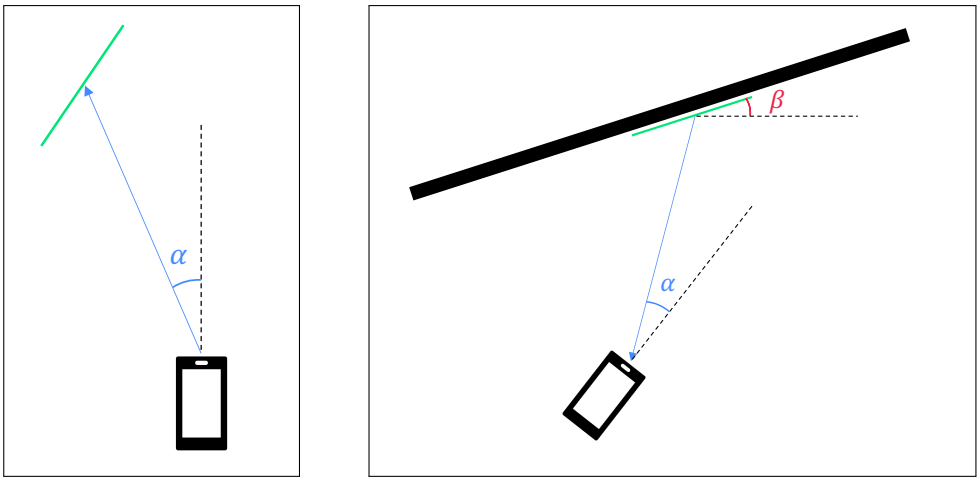
In this section, we explain the implementation approach for localizing the user within the museum. We detail the procedure for calculating the position while tracking an artwork and introduce the sensors utilized for localization when no artwork is in proximity. We outline two techniques for localization, namely dead reckoning and PDR. We observed that PDR outperforms dead reckoning in the museum setting, which is explained in detail in Section 5.1.4.

4.3.1. Artwork-based Position Estimation

Localization accuracy is crucial in the navigation app and tracking an artwork provides an accurate estimate of the user's position. Such an estimate is necessary for initializing the localization when starting the navigation app. It is also used to correct any errors that may have been introduced in the intermediate position estimation while not tracking any artworks.

Vuforia provides the relative position and orientation of the artwork with respect to the camera used to track it, as illustrated in Figure 4.4 (a). Since the absolute position of the artwork is already known from the floor plan editor, the relative position is added to it while taking the orientation of the artwork into account.

It is worth mentioning that the position and orientation of the artwork is provided by Vuforia with the assumption that the camera is located at position $(0, 0, 0)$ and with no rotation. Therefore, to calculate the position estimate, we apply the reverse rotation α to the user's position, in addition to the real-world rotation β of the artwork. This process is illustrated in Figure 4.4 (b).



(a) Data obtained via tracking

(b) Calculation of position estimate

Figure 4.4.: *Explanation of artwork-based position estimation*

Smoothed Tracking

The tracking data obtained from Vuforia is noisy, especially when the user stands far away from the artwork. To reduce jittery tracking data, the system applies a smoothing technique by calculating a moving average of the artwork's position and orientation. The window size is set to 20 frames, which corresponds to 0.4 seconds. The user's position is only updated when the window is full, which improves the reliability of the position estimate. We show the pseudo code of the smoothing algorithm in Alg. (1). The code is executed in each frame. In the algorithm, p_t is the tracked position, p_s is the resulting smoothed position, w is the window size, and $window$ is a queue that contains all positions within the window. The smoothing works analogously for the orientation.

Algorithm 1 Moving Average

$window.Enqueue(p_t)$
if $window.Length == w + 1$ **then**
  $p_s \leftarrow p_s + 1/w * (p_t - window.Dequeue())$
else
  $p_s \leftarrow p_s + p_t$
  if $window.Length == w$ **then**
    $p_s \leftarrow p_s/w$
  end if
end if

The effect of the smoothing technique is depicted in Figure 4.5. When the user is close to the artwork, the impact of smoothing is relatively modest. The raw data are accurate enough so that smoothing is not strictly necessary. However, at a greater distance from the artwork, the unprocessed data display outliers that substantially deviate from the true position. The smoothing process considerably diminishes the influence of these outliers.

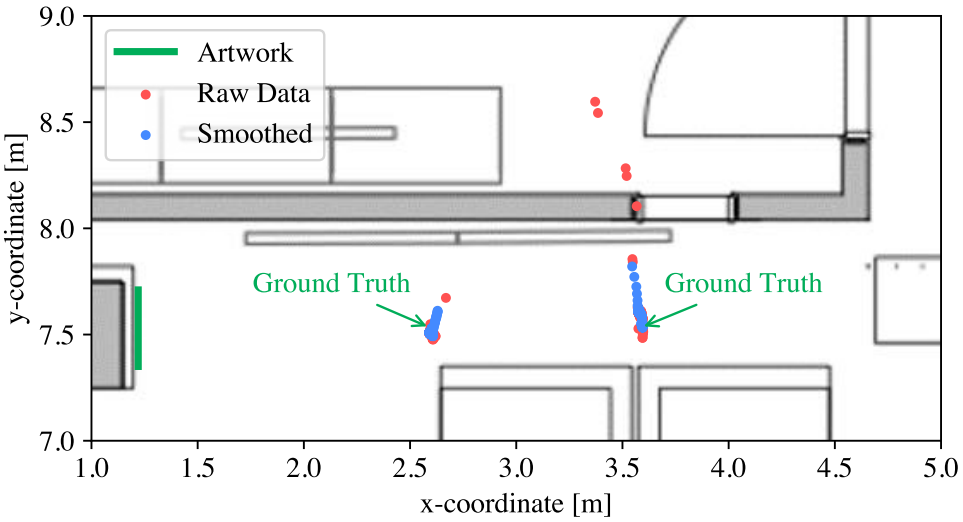


Figure 4.5.: *Artwork-based position estimate with and without smoothing*

Rapid Turns

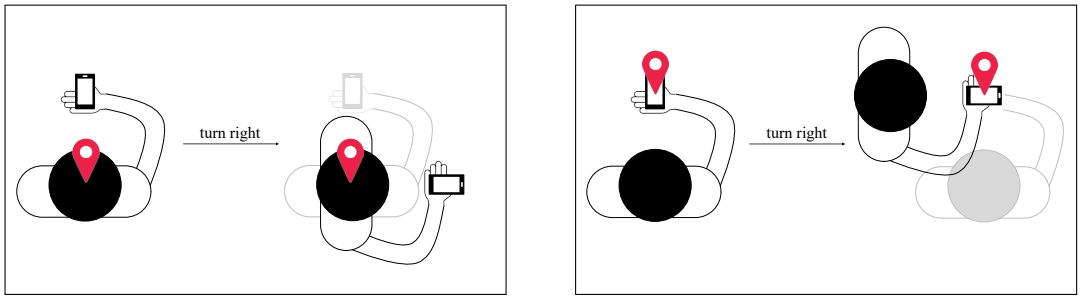
The application of smoothing to the orientation gives rise to a new issue, namely a delay in the orientation estimate when users turn rapidly. The smoothing process involves averaging the orientation over a time window, resulting in a delay between the estimate and the actual orientation. This delay is proportional to the turning speed. The issue is particularly evident when the user turns away from an artwork and stops tracking it. As the last orientation estimate is used as the initial value for the sensor updates and the estimate lags behind the actual orientation, this situation results in an erroneous orientation estimate after this point in time.

To mitigate this problem, we adopt a strategy of switching to the orientation calculated via gyroscope data as soon as the user starts turning rapidly. To detect such turns, we set a threshold

on the gyroscope data. The threshold was set to $1°$ per frame, which corresponds to $1°$ per 0.02 seconds. This method effectively overcomes the delay issue.

Phone Position

An important factor that can be easily overlooked is the distance between a user's body and their phone. Typically, the phone is held at a distance of approximately 30-40 cm from the body. Vuforia provides information about the phone's position relative to the artwork, resulting in an estimate of the phone's position, not the user's body. The issue becomes evident when a user turns in place. Using the phone's position results in an incorrect interpretation of the user's turn. Specifically, it suggests that the user is turning around the phone rather than the phone turning around the user. This issue is illustrated in Figure 4.6, with a visual representation of the user seen from above. Placing the estimate at the user's body, as shown in Figure 4.6 (a), better models a user's natural behavior than the location shown in Figure 4.6 (b).



(a) Estimate located at body **(b)** Estimate located at phone

Figure 4.6.: *Implications of different estimate locations*

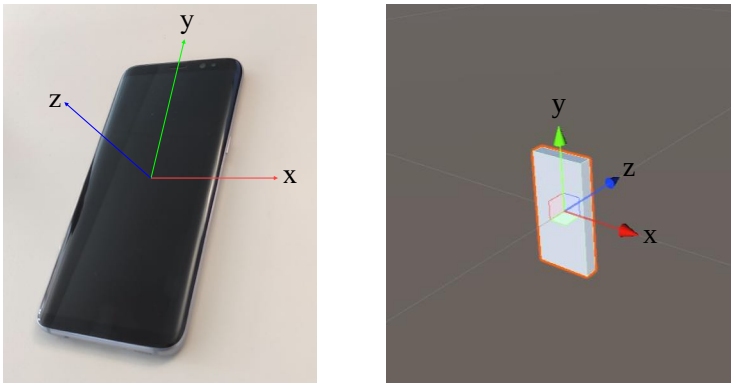
To place the estimate at the user's body, the distance between the phone and the user's body is added to the distance obtained from Vuforia. Assuming that the user holds the phone in front of their body, this additional offset will result in a more precise estimation of the user's position. This adjustment does not alter the navigation process; rather, it places the position estimate closer to where one would expect it to be.

4.3.2. Sensor Data

This section delineates the process of reading and processing sensor data. This process happens prior to feeding the data into the distinct localization implementations.

The accelerometer provides information on the distance covered by the user, while the gyroscope gives data on the user's orientation. The acceleration data are presented in g-force values, so the values must be multiplied by 9.8 to obtain the acceleration in m/s^2. The gyroscope provides the angular velocity in rad/s, so we multiply it with $\frac{180}{\pi}$ to get $°/s$.

As the phone's coordinate system is different from the one used in Unity, it is necessary to multiply the z-axis data by -1. This adjustment is done for both the accelerometer data and the gyroscope data. The two coordinate systems are visualized in Figure 4.7.



(a) Phone **(b)** Unity

Figure 4.7.: *Three axes in the phone's and Unity's coordinate systems*

It is worth noting that the acceleration data are inevitably influenced by gravity. However, reading sensor data through Unity gives access to the linear acceleration, which corresponds to the acceleration without the effect of gravity. Throughout our implementation, we therefore refer to the acceleration that is not affected by gravity.

To reduce the impact of noise in the sensor data, a low-pass filter is utilized. In Unity, this filter is implemented by linearly interpolating between the new value and the previous filtered value. The interpolation calculation is defined in Eq. (4.1), where a is the already filtered value, b is the current noisy value, and p is the interpolation parameter. The specific value of $p = 0.4$ was determined through experimentation. Adjusting p has an impact on the noise level in the filtered data; higher values leave more noise, while lower values lead to loss of certain details present in the initial data. The value resulting from this calculation is utilized as a in the subsequent frame.

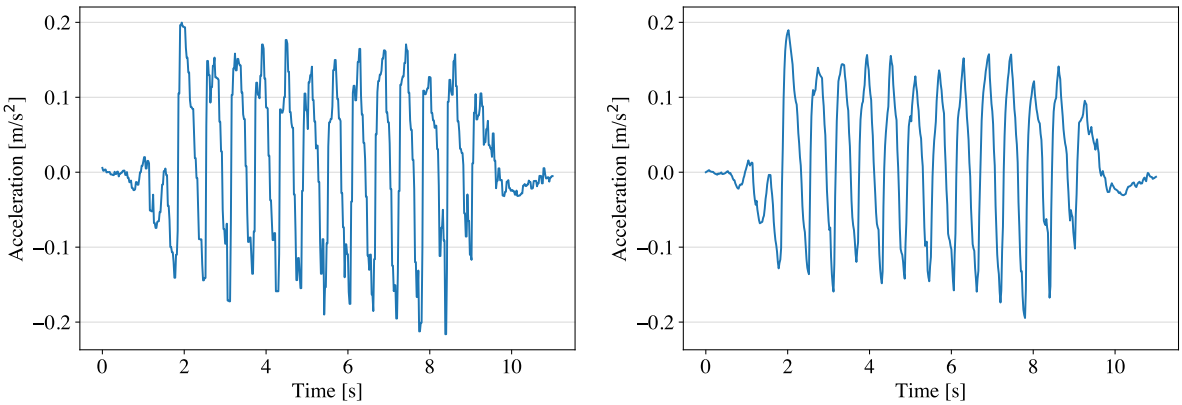
$$a + (b - a) * p \tag{4.1}$$

The outcome of the filtering process is visualized in Figure 4.8. The data were recorded when walking down a straight corridor with constant speed. It is evident that the filtered data are smoother, while still retaining the fundamental trends observed in the original data. The results indicate that setting $p = 0.4$ strikes a favorable balance between noise reduction and preservation of data details.

The application of a low-pass filter is limited to the acceleration data and not extended to the data obtained from the gyroscope. We found that filtering the gyroscope data was not necessary, as discussed in further detail in Section 5.1.1.

4.3.3. Dead Reckoning Implementation

The implementation of dead reckoning is based on data from the accelerometer and the gyroscope sensors. In the following subsections, we provide details on the implementation of the different features.



(a) Unfiltered **(b)** Filtered

Figure 4.8.: *Acceleration in local y-axis with and without filtering*

Integration

The dead reckoning position updates are computed by integrating the sensor data. To obtain an accurate position estimate, it is essential to calculate both the position and orientation of the user. The position p is derived from integrating the acceleration a twice, while the orientation o is derived from the integration of the angular velocity ω. This computation is shown in Eq. (4.2), where v is the velocity and t is the elapsed time.

$$\begin{aligned} p &= \int v\,dt = \iint a\,dt \\ o &= \int \omega\,dt \end{aligned} \tag{4.2}$$

The primary limitation of the integration-based position updates is the accumulation of errors over time. Integrating the sensor data amplifies any error that may exist. This phenomenon is shown in Eq. (4.3), where the error e in the sensor data is integrated and subsequently manifests in the position estimate. Without compensating for these inaccuracies, the error will continue to increase over time.

$$\begin{aligned} \iint a{+}e_a\,dt = \int v{+}e_a t\,dt = p{+}\frac{e_a t^2}{2} \\ \int \omega{+}e_\omega\,dt = o{+}e_\omega t \end{aligned} \tag{4.3}$$

Calibration

One strategy to mitigate the issue of erroneous data is to calibrate the device when starting the navigation app. The calibration process aims to eliminate the deviation of data from its true

value, also known as drift. Specifically, when the phone is positioned on a stable surface, the sensor should output an acceleration and an orientation change of zero. Any value that differs from this expected value is regarded as drift.

Calibrating the phone involves placing it on a stable surface and calculating the average value over a period of 1 second. Averaging over this duration minimizes any remaining noise in the data. The resulting average value reflects the drift inherent to the sensor. Subsequently, the drift is continuously subtracted from all sensor readings to ensure that the values of a stationary device are indeed zero.

Zero Velocity Potential Update

In addition to the aforementioned technique, ZUPT can also be utilized to enhance the accuracy of localization. It operates by detecting whether the user is stationary and resets the current velocity to zero if that is the case. This procedure effectively stops the accumulation of the integration error, leading to improved accuracy.

The implementation of ZUPT involves comparing the current acceleration and change in orientation with predefined thresholds. If the values of both the acceleration and change in orientation are smaller than their respective thresholds, the ZUPT conditions are met, and the current velocity is reset to zero. The thresholds are based on previous research conducted by Suresh and colleagues [SSPT18] and are presented in Section A.1.

In our implementation, the velocity is reset only if the ZUPT conditions are met for a minimum of five consecutive frames, which corresponds to 0.1 seconds. This approach eliminates false positives that can occur when the conditions are met but the user is actually walking. Resetting the velocity while the user is walking can have detrimental effects on the position estimate, as it may cause the estimate to fall behind the true position and never catch up again. Our own experiments demonstrated that a window of five consecutive frames is sufficient to mitigate the occurrence of false positives.

Although the ZUPT method is effective in preventing the accumulation of errors during periods of standstill, it does not eliminate any errors that have already accumulated. As a result, its ability to improve localization accuracy is limited to stationary periods. During periods of movement, errors still accumulate and affect the accuracy of the position estimates.

4.3.4. Pedestrian Dead Reckoning Implementation

The implementation of PDR consists of three distinct parts: step detection, step length estimation, and heading estimation. Together, these parts provide an update from the current position to the position after a step. The implementation utilizes data collected from both the accelerometer and the gyroscope of the mobile device.

Step Detection

The step detection algorithm relies on acceleration data and utilizes prior knowledge of the user's orientation obtained from tracking data, which indicates the direction of gravity.

A step can be defined as a temporal sequence characterized by an initial upward movement followed by a downward movement of the body. This sequential pattern, referred to as bounce, was first introduced by Weinberg [Wei02] and is visually depicted in Figure 4.9. The figure illustrates a single step, capturing key positions. At the beginning, both legs are in contact with the floor, with the right leg positioned in the back. During this phase, the hip is at its lowest point in the trajectory. As the step progresses, the hip ascends to its maximal elevation when the left leg aligns directly beneath it. Bringing the right leg forward and to the ground leads to a descent of the hip. The vertical displacement of the hip throughout this sequence is referred to as bounce.

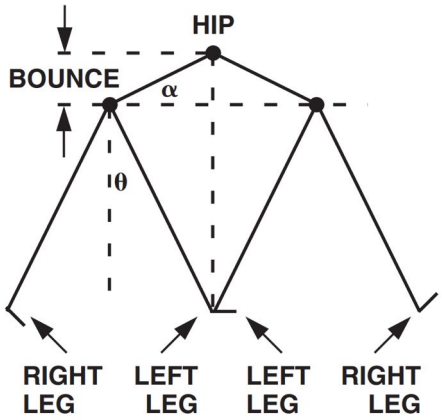


Figure 4.9.: *Bounce while walking*

The step detection algorithm detects the bounce by analyzing the acceleration along the direction of gravity. Since we assume that the floor plan lies flat on the xz-plane in Unity, the direction of gravity corresponds to the y-axis. Therefore, the relevant acceleration data are denoted by a_y.

The human body exhibits a positive peak in acceleration while going up and a negative peak while going down. The acceleration during a step is depicted in Figure 4.10. The detection of a step begins as soon as the acceleration a_y exceeds a predefined step detection threshold. For this application, the threshold is set at $T_{step} = 0.7m/s^2$. Two helper variables, *goingDown* and *comingBackUp*, are used to determine the current stage of the step detection. *GoingDown* is set to true if a step has been started and the acceleration is less than the threshold T_{step}. *ComingBackUp* is set if *goingDown* is true and $a_y < -T_{step}$. During the step, a_{max} and a_{min} are constantly updated. Once both *goingDown* and *comingBackUp* are true and $a_y > -T_{step}$, the algorithm confirms the detection of a step. In Figure 4.10, the detection is indicated by "End of Step".

Information regarding the detected step is retained in a class called *StepHelper*. This class includes the orientation of the user at the start and at the end of the step, maximum and minimum

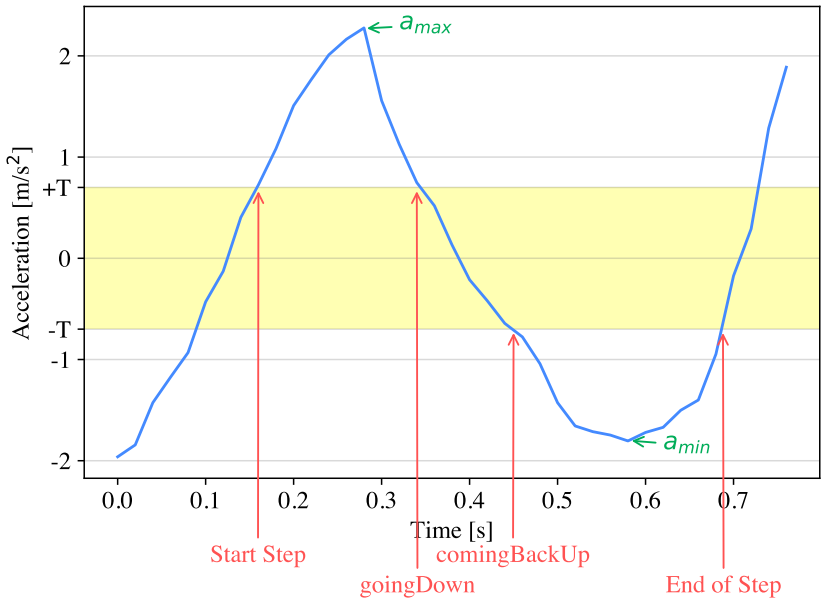


Figure 4.10.: *Process of step detection using the vertical acceleration*

acceleration during the step a_{max} and a_{min}, and the duration of the step. The duration is used to disregard any steps that take longer than a designated maximum step duration. We set this maximum value to 1.2 seconds. If the duration of a step exceeds this limit, it is presumed to be an anomalous reading and the step is discarded.

During the experiments conducted in the development of the localization algorithm, we identified the issue that tapping on the screen can erroneously be detected as a step, leading to incorrect position updates. To address this issue, we introduce a horizontal threshold which is combined with the step detection threshold. Specifically, a step detection cycle is initiated only when $a_y > T_{step}$ and the magnitude of the horizontal acceleration, consisting of a_x and a_z, exceeds the horizontal threshold T_h. We set $T_h = 0.55m/s^2$. This constraint is mathematically expressed in Eq. (4.4). It effectively prevents screen taps from being detected as steps, while retaining the detection of true physical movements.

$$a_y > T_{step} \wedge \left| \begin{pmatrix} a_x \\ a_z \end{pmatrix} \right| > T_h \tag{4.4}$$

Step Length Estimation

The step length estimation can be done either dynamically or statically. The dynamic approach involves computing the step length for each step individually, while the static approach employs a fixed step length for all steps.

The dynamic step length estimation we use is based on research conducted by Weinberg [Wei02]. He shows that the step length estimate l as shown in Eq. (4.5) has an accuracy of $\pm 8\%$. The calculation involves the minimum and maximum values of the acceleration during the step, which

are stored in the *StepHelper* class.

$$l = \sqrt[4]{a_{max} - a_{min}} \tag{4.5}$$

This estimate was originally designed for use with pedometers worn on the foot. When using smartphones, the estimate may be affected by additional noise introduced by the user's hand movements while holding the device. .

The static step length estimation is a non-adaptive approach that utilizes a fixed value for step length. This approach is independent of the way the user holds the phone during walking. The step length is personalized to each individual by measuring their average step length over a short distance and using that value for all detected steps.

Heading Estimation

The heading estimation process utilizes the orientation obtained by integrating the angular velocity. Specifically, the *StepHelper* class stores the orientation of the user at the beginning and at the end of each step. The direction of the step is determined by calculating the angle between those two orientations.

When a user takes a step, it has no effect on the vertical component of the position estimate. As a result, the heading estimation requires only the angle around the world's y-axis to determine the direction of the step. This angle is derived by considering the direction in which the phone's camera is oriented, corresponding to its local z-axis. The angle of the local z-axis around the world's y-axis is calculated by discarding the y component of the local axis and applying trigonometric principles to the remaining vector. The resulting angle is used as heading estimate.

4.4. Pathfinding Implementation

All functionality related to pathfinding was implemented using Unity and compiled to run as an Android app. First, the pathfinding algorithm triangulates the polygons that were defined in the floor plan editor. Next, the triangulated mesh is utilized to generate the NavMesh. The NavMesh is then responsible for computing the path to the destination. These steps are further elaborated below.

4.4.1. Triangulation

The polygons stored in the database are represented as ordered lists of vertices that follow the polygon border. To create a NavMesh, Unity requires a mesh. In order to create a mesh from these polygons, they need to be triangulated. We employ the ear clipping algorithm as explained by David Eberly [Ebe08] to triangulate the polygons.

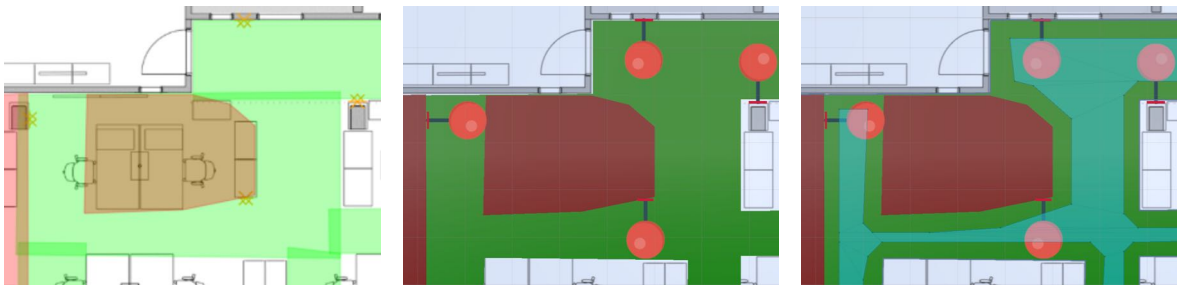
The ear clipping algorithm assumes that the input is a simple polygon. In a simple polygon, each vertex has exactly two edges and edges may only intersect at vertices. Although this constraint is currently not enforced in the floor plan editor, creating non-simple polygons would result in an unintuitive description of the walkable area. Therefore, it is assumed that the user only defines simple polygons in the floor plan editor. Only then does the ear clipping algorithm function properly.

The basic idea of the ear clipping algorithm is to find "ears" of the polygon, which are convex vertices that can be clipped off by creating a triangle with its two adjacent vertices. This process is repeated until all vertices have been incorporated into triangles, resulting in a valid triangulation. The algorithm has a time complexity of $O(n^2)$ for n vertices and is widely used in computer graphics and computational geometry applications.

Our implementation of the ear clipping algorithm was adapted from an online tutorial[7]. We made several modifications to customize it to our specific application and addressed certain bugs in the original implementation.

4.4.2. NavMesh

The NavMesh creation process comprises three distinct stages. Initially, the system starts with a set of polygons defined in the floor plan editor, as illustrated in Figure 4.11 (a). The triangulation leads to a mesh for each polygon. These meshes are then integrated into the scene in Unity, as illustrated in Figure 4.11 (b). They are used to generate the NavMesh. The process of generating a NavMesh is referred to as baking. The final result of the baking process is shown in Figure 4.11 (c).



(a) Polygons in floor plan editor **(b)** Triangulated polygons in Unity **(c)** Baked NavMesh

Figure 4.11.: *Steps taken to create the NavMesh*

To generate the NavMesh, it is essential to define the type of agent that will navigate the environment. Defining the agent type is done before baking the NavMesh. A NavMesh agent is represented as a cylinder that in our case resembles a person. The height of the cylinder is not a crucial factor in our application. The radius is set to 30 cm, which approximately corresponds to the space occupied by a human. The agent's radius determines the distance between the edges of the NavMesh and the edges of the actual walkable area. It can be interpreted as the

[7] `https://www.habrador.com/tutorials/math/10-triangulation/` (Accessed: 24. May 2023

minimum space required to allow the agent to walk on the NavMesh and always keeping the whole cylinder within the walkable area. Therefore, the NavMesh depicted in Figure 4.11 (c) is smaller than the walkable area represented by the green mesh below it.

4.4.3. Calculating a Path

A NavMesh agent possesses the capability to autonomously calculate its own path. The process of pathfinding is based on an implementation of the A*-algorithm. When the user specifies the desired artwork, the system sets the agent's destination. Subsequently, Unity computes the optimal path between the agent's current position and the destination. The output of the A* calculation is a path consisting of the start and end points, along with intermediate vertices. We refer to all vertices of the path as corners.

4.5. Methods to Display Directions

This section covers the implementation details of the three methods of displaying navigation information. These methods are based on the path calculated as presented in the previous section.

Locating the instructions that indicate the direction to proceed may be challenging, particularly when they falls outside the visible screen area. To address this challenge, we implement a visual cue. This cue serves as a helpful indicator of the instructions' location when they are not currently within the user's field of view. By providing such a hint, users can easily identify the position of the instructions, enabling them to navigate to the intended direction. The visual cue is depicted in Figure 4.12 and is available in all three methods.

4.5.1. *Fixed*

The *fixed* method involves presenting an arrow at a fixed distance of 1.3 meters from the user, as shown in Figure 4.13. The arrow's position is determined by following the path to the destination for a distance of 1.3 meters. Following the path may include a straight line from the user or it may include turning around a corner. Consequently, the arrow may not always be positioned directly in front of the user but could appear on their right or left, depending on the path's geometry. The arrow's angle is determined by the angle of the sub-path it is located on. This implementation enables the user to follow the path by consistently moving towards the position of the arrow. Its design is straightforward and intuitive, minimizing the cognitive load of the user.

4.5.2. *Corner*

The *corner* method places the arrow at the next corner of the path. As illustrated in Figure 4.14, the arrow is oriented the direction of the sub-path that follows that corner. If the next corner is the destination, the arrow points directly towards the artwork. In practice, we do not use the

Figure 4.12.: *The visual cue indicating the instructions' location*

Figure 4.13.: *The fixed method*

next corner of the path, but the last one visible from the user's current position. The reason for this distinction has to do with the way the path goes around corners.

Placing the arrow at the first corner of the path, as shown in Figure 4.15 (a), does not provide an accurate indication of the direction in which the path continues. The arrow's orientation is misleading and may result in confusion for the user. The issue can be addressed by placing the arrow at the last visible corner, as shown in Figure 4.15 (b), which more accurately reflects the direction that the user should take. This example highlights the importance of placing the arrow at the last corner that is visible to the user, rather than simply at the next corner in the path.

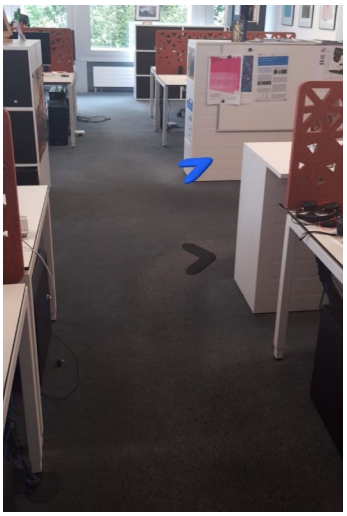
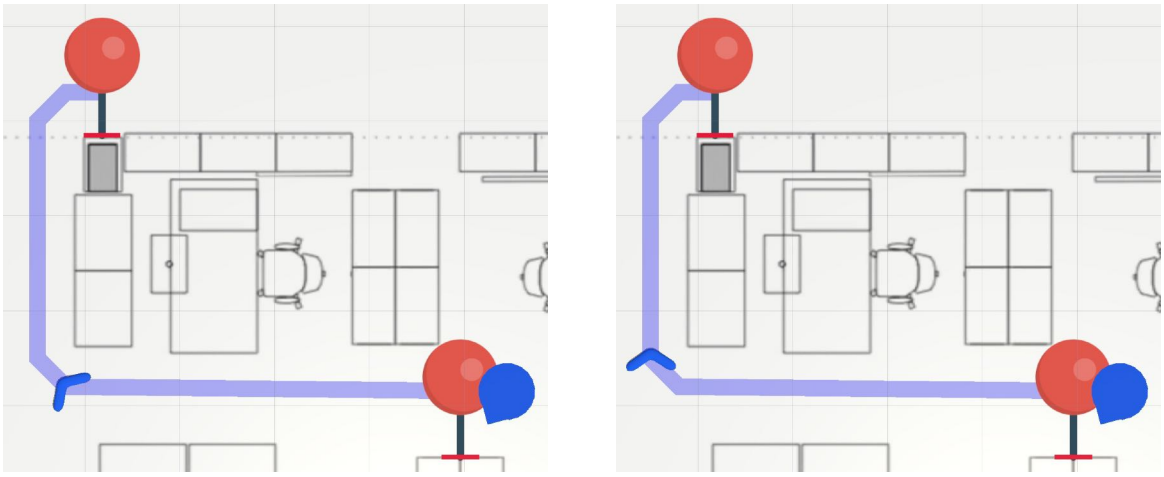


Figure 4.14.: *The corner method*

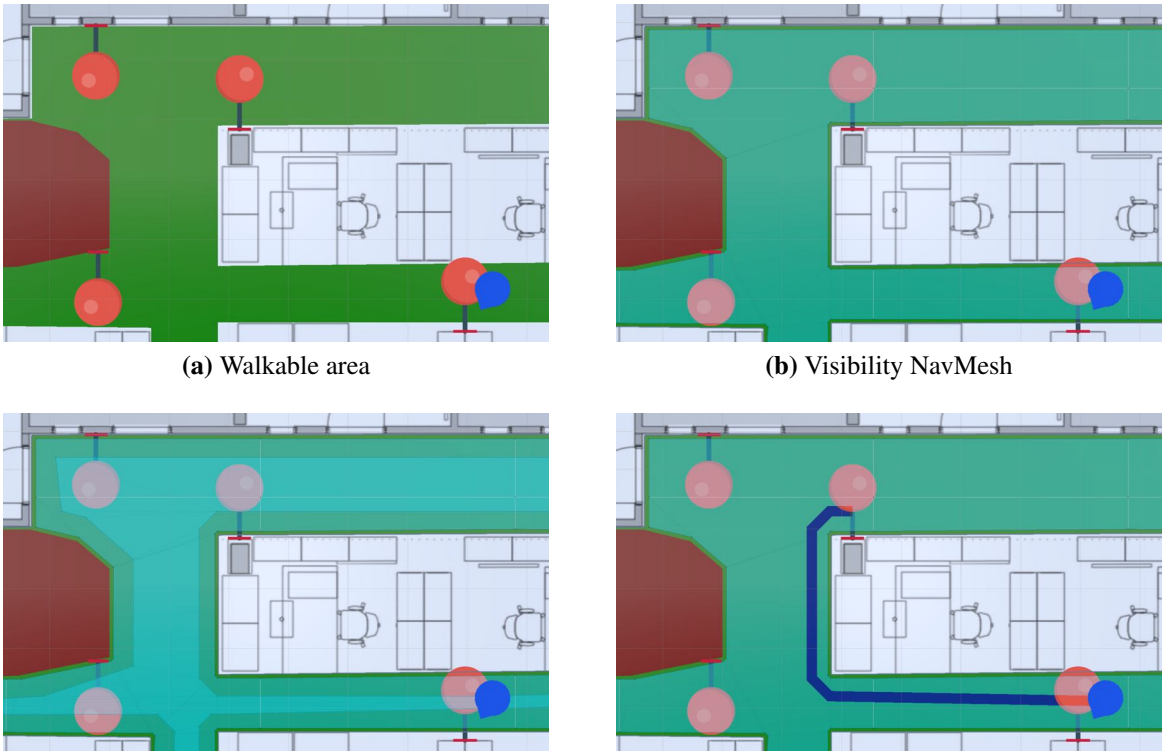


(a) At first corner **(b)** At last visible corner

Figure 4.15.: *Placement of the arrow along the path with the corner method*

To determine whether a corner is visible to the user, we introduce another NavMesh, referred to as the ”visibility NavMesh”. Unlike the normal NavMesh, the visibility NavMesh uses an agent with a radius of 5 cm and therefore represents the walkable area more closely than the normal NavMesh. A comparison of the walkable area and the visibility NavMesh is given in Figure 4.16 (a) and (b). The normal NavMesh is shown on top of the visibility NavMesh in Figure 4.16 (c) to clearly indicate their difference. The calculated path is shown in relation to the visibility NavMesh in Figure 4.16 (d). We see that in corner regions, the trajectory of the path aligns closely with the edges of the normal NavMesh. This alignment leads to the presence of adjacent vertices in close proximity along the path. Therefore, it is essential to consider the last visible corner instead of the next corner along the path to ensure accurate placement of the arrow.

Determining whether the user sees a specific point on the floor plan from their current position can be achieved by calculating the path to that point using the visibility NavMesh. If the



(a) Walkable area

(b) Visibility NavMesh

(c) Normal NavMesh

(d) Calculated Path

Figure 4.16.: *Walkable area and NavMeshes*

resulting path consists of a single straight line, it indicates that the point is visible from the user's location. However, this approach is associated with the limitation that we have to wait for the completion of the NavMesh path calculation. The duration of the calculation cannot be predicted in advance, rendering this approach less favorable.

Instead, we employ a custom algorithm to conduct the visibility check. This algorithm initiates at the specific point of interest and takes incremental steps towards the user's position. At each step, the algorithm evaluates whether the point falls within the boundaries of the visibility NavMesh. For a point to be deemed visible to the user, all the intermediate steps between that point and the user's position must lie on the visibility NavMesh.

To check whether a point is on the visibility NavMesh, we make use of the *SamplePosition* function provided by the NavMesh. This function returns the closest NavMesh point within a specified radius around the given input point. In our application, we use a radius of 5 cm to compensate for the small difference between the size of the visibility NavMesh and the defined walkable area. Thus, the function returning a point indicates that the input point is within the walkable area.

With this functionality, the algorithm iterates through the corners of the path and determines the last one that is still visible to the user. At that corner, the arrow for the *corner* method is placed. It is important to note that the last visible corner may or may not be the first corner of the path.

4.5.3. *Many*

The default method used in the app is *many*, which is shown in Figure 4.17. In this method, several arrows are positioned with a fixed distance of 50 cm between them. The placement of the arrows is based on their distance from the destination artwork, not the distance from the user's position. The last arrow is placed 50 cm away from the destination artwork. This design ensures that the arrows remain at the same spatial location even when the user moves, resulting in a more intuitive navigation experience.

Figure 4.17.: *The many method*

The algorithm for the *many* method also utilizes the visibility NavMesh to determine the last visible corner along the path. Prior to this corner, the entire path is visible and arrows are placed accordingly. Beyond this point, the algorithm checks each potential arrow position to determine if it is still visible to the user. It is possible for a position to be visible to the user despite being beyond the last visible corner. Once the last visible arrow position is determined, the algorithm places two additional arrows at their respective positions, even if they are technically no longer visible. Even though the view without occlusions is not an accurate representation of reality, we find that the inclusion of additional arrows enhances user comprehension of the given directions.

5

Evaluation

In this section, we present the results of our empirical investigation of the implemented application and discuss the implications derived from the findings. We begin by providing an analysis of the reliability of the sensor data collected for localization. Subsequently, we describe the user study conducted and the deductions drawn from the results.

5.1. Sensor Data

This section describes all results related to the sensor data, including the two localization approaches dead reckoning and PDR. We also describe various experiments and findings we have made on sensor accuracy and drift.

5.1.1. Accelerometer vs. Gyroscope

In the implementation of dead reckoning, we observed that the acceleration data were considerably noisy and susceptible to drift. In contrast, the data obtained from the gyroscope were comparatively less problematic and provided accurate estimates of the user's orientation over time. Considering these observations, we conclude that it is necessary to perform filtering and calibration exclusively on the acceleration values.

5.1.2. Dead Reckoning Results

The implementation of dead reckoning was observed to exhibit large drift in accelerometer readings, resulting in a substantial discrepancy between position estimates and actual positions. This

discrepancy was evident even when the phone was statically placed on a table. Specifically, our experiments showed that after just 10 seconds at rest, the estimated position was already more than 70 centimeters away from the true position, as depicted with the blue line in Figure 5.1. Furthermore, the error accumulation became increasingly rapid, resulting in a 14 meter offset after 40 seconds. These results suggest that the dead reckoning implementation may not be suitable for applications that require high-precision localization over extended periods of time.

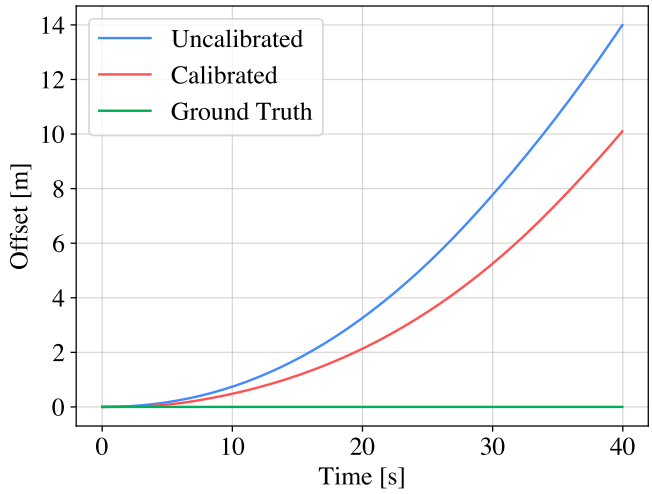


Figure 5.1.: *Offset of dead reckoning position estimate with phone at rest*

In the implementation of dead reckoning, we keep track of the current velocity as there is minimal acceleration during steady walking. When the magnitude of the drift surpasses the actual acceleration during constant walking, it influences the current velocity and deviates it from the correct direction. The results of this effect are shown in Figure 5.2, where a user walked along a straight line with constant speed. We show the results of five such walks. In the first few moments, the initial acceleration of starting the motion contributes to an accurate update of the position. Then, the drift gradually becomes more prominent and negatively impacts the accuracy of the position estimate as the walk progresses.

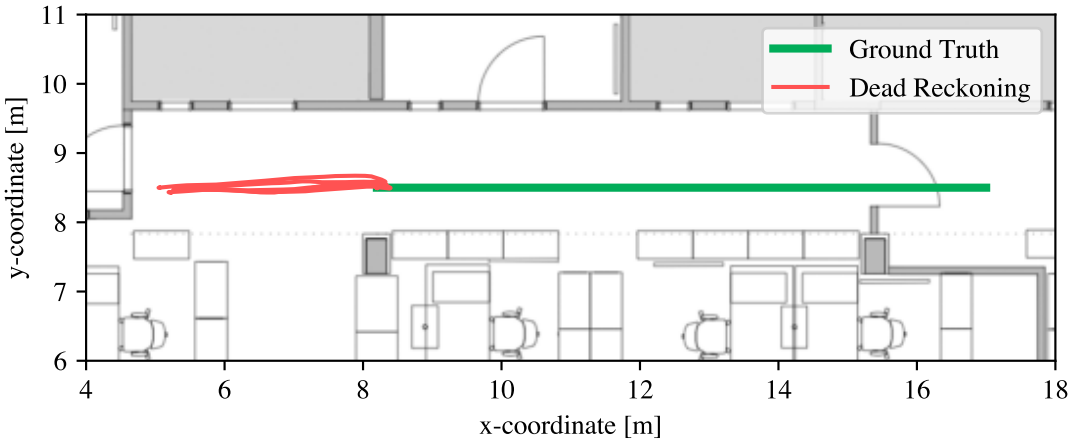


Figure 5.2.: *Results of dead reckoning localization*

The results depict the orientation of the drift as being directed backwards in the local coordinate system of the device. As a result, the current velocity also points backward, leading to a substantial deviation of the position estimate in direction opposite of the true walking direction.

Extensive testing was conducted on the dead reckoning implementation, ruling out any potential coding errors. The tests confirmed the correctness of the implementation. When drift is present in the acceleration, it gets integrated and added to the current velocity, resulting in a continuous increase in velocity due to the drift. This effect, already seen in Figure 5.1, is also evident in Figure 5.3, where points were marked at regular intervals of 0.4 seconds as a person walked to the right. The increasing distances between these points over time indicate that these updates do not reflect the correct position updates. In correct updates, the distances between the points would remain the same since the experiment was conducted with a constant walking speed. The observed growing distance provide further evidence that drift is present in the acceleration.

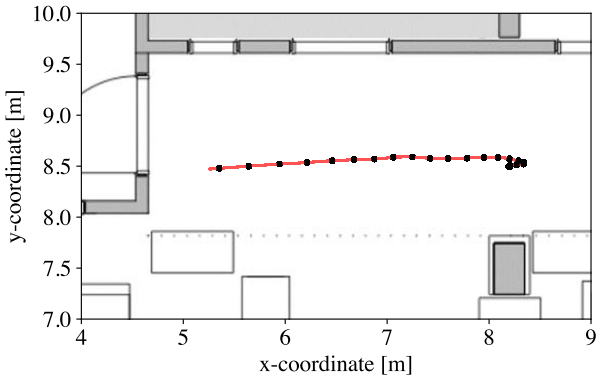


Figure 5.3.: *Dead reckoning results with a point every* 0.4 *seconds*

Figure 5.3 provides further evidence of the correctness of the implementation by demonstrating that the initial movement of the position estimate aligns with the correct direction. This observation serves as an additional indicator that the axes were not flipped incorrectly, as the entire movement would be in the wrong direction if such a misalignment existed. The initial movement in the correct direction confirms that the accumulated error, resulting from drift, is the main cause of deviation.

The same effect is illustrated in Figure 5.4, where we conducted a walk covering the same distance as before at different speeds. The results illustrate that the portion of the path aligned with the correct direction increases with walking speed. A higher initial acceleration during the walk leads to a longer duration in which the walking acceleration surpasses the drift, resulting in a delayed deviation in the position estimate.

Although the direction of drift may be unique to the smartphone utilized in our experiments, the strength of the drift highlights the considerable impediment it poses to precise localization. This challenge is particularly relevant for applications that aim to be compatible with a diverse range of devices.

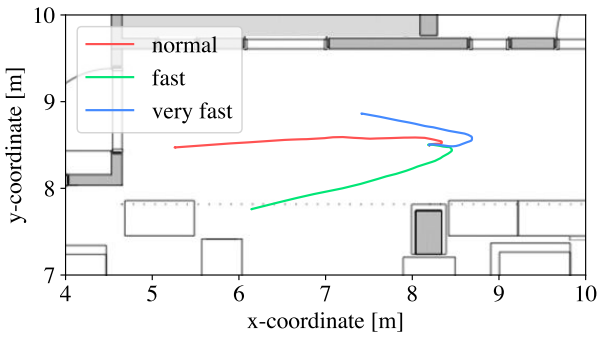


Figure 5.4.: *Results of dead reckoning with different walking speeds*

Calibration Results

We performed a calibration of acceleration values to mitigate the effects of drift. However, the results were inconclusive. Similarly to previous experiments, evaluations were conducted on the phone at rest. The results are presented as the orange line in Figure 5.1. The estimated position was found to be closer to the true value compared to uncalibrated estimates. Nevertheless, over time, the position estimate deviated from the actual value even after calibration. Despite the reduction in drift after calibration, the offset still exhibits a substantial rate of growth over time, indicating that the effectiveness of the calibration procedure in mitigating the drift is limited.

The elimination of drift through calibration presents inherent challenges due to the persistent noise in the acceleration data, even after filtering. This phenomenon is demonstrated in Figure 5.5, where the calibration process succeeds in bringing the acceleration values closer to zero but residual noise remains. The average value fails to precisely eliminate the drift, as it does not represent its true magnitude. Figure 5.5 also illustrate how the drift is not constant but varies over time, making it impossible to eliminate.

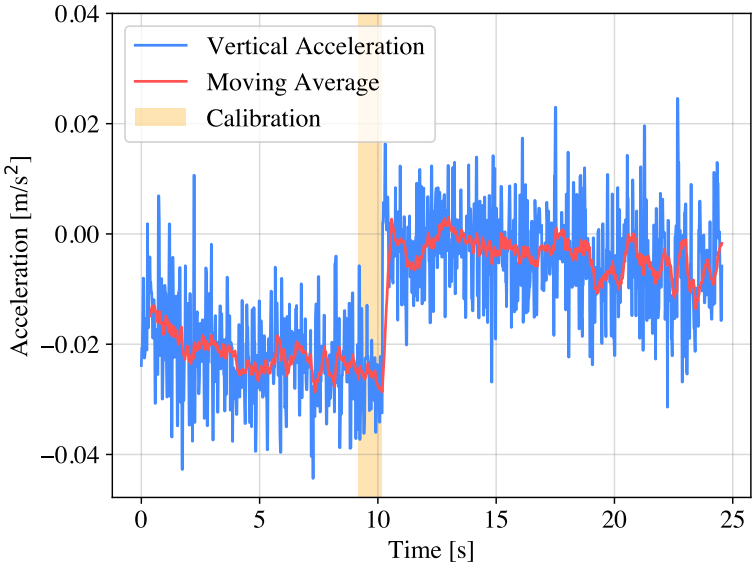


Figure 5.5.: *Calibration of vertical acceleration*

Zero Velocity Potential Update Results

The primary objective of ZUPT is to stop the accumulation of errors. It is not capable of eliminating the already accumulated error, but it can prevent further deviation from the current position when the user is at rest.

The ZUPT algorithm performs effectively in detecting when the user is stationary, resulting in an improvement in the accuracy of the dead reckoning method during stop-and-go motion. However, for extended periods of continuous motion, ZUPT cannot provide any improvement in the position estimate. In these cases, the accumulated error remains a challenge to be addressed.

An Examination of Drift

We conduct an in-depth analysis of drift on the Galaxy S8 smartphone used in this thesis. The objective is to investigate potential factors that could contribute to changes in average acceleration values. Our aim is to determine whether the drift is influenced by fixed factors, such as position or temperature.

To investigate the drift, we calculate the average acceleration at rest over a two-second period to minimize the effects of noise present in the sensor readings. To determine the extent to which the drift is affected by different factors, we conduct a series of experiments. The first experiment involves measuring the drift 10 times and moving the phone to a different position for each measurement. In the second experiment, we flip the phone $180°$ along each of its axes to determine whether the drift is influenced by the phone's orientation. Finally, we leave the phone at rest for a period of 10 minutes while continuously measuring the drift to investigate whether it is affected by changes in the phone's temperature, which can increase over time.

In all conducted experiments, it was observed that the drift exhibited a relatively stable behavior across multiple measurements. This result suggests that the drift is not considerably influenced by factors such as position, rotation, or temperature variations. However, the drift did exhibit fluctuations between two distinct and relatively stable levels. The exact triggers or causes for the transition between these levels could not be identified reliably. At times, the lower level of drift was observed in a particular measurement, but upon restarting the application without any physical movement of the phone, the drift level would shift to the higher level. Similarly, changes in the drift level were occasionally observed after walking and tracking artworks for some time. However, it was not possible to consistently reproduce these results under controlled conditions. These variations highlight the highly unpredictable characteristics of the drift.

Figure 5.6 shows the drift for each of the three axes over fourteen minutes. The drift was measured every 6 seconds with the "Drift Check" option from the settings page. We clearly see that there is noise present in the data, even though the phone is at rest and the data points represent averages over 2 seconds. The variability of the measured drift makes it hard to eliminate, as there is always a remaining error. At about 13 minutes, the drift jumps to the higher level. At that point, a charger was plugged into the phone. This jump indicates that charging has some influence on the phone which changes the acceleration values. Although this scenario is unlikely to happen during a museum visit, it again shows the unpredictability of sensor data.

In conclusion, the presence of drift gives rise to two primary challenges. Firstly, despite at-

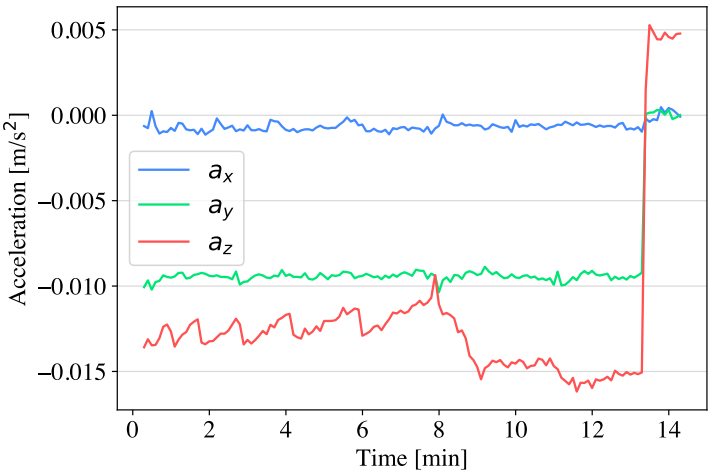


Figure 5.6.: *Drift measured over long period of time*

tempts to mitigate its impact, the drift component remains noisy, resulting in varying average values over time. Consequently, accurately determining the true value of the drift becomes an impossible task. Secondly, the highly unpredictable nature of the drift presents a substantial obstacle. The two identified levels of drift do not exhibit a discernible trigger or pattern that can reliably indicate a transition between them. Consequently, relying on a single value to effectively counteract drift becomes futile, as that value would be inaccurate when the drift level undergoes a change.

5.1.3. Pedestrian Dead Reckoning Results

PDR is not subject to drift as it does not use absolute acceleration values in its calculations. In order to detect steps, the acceleration must exceed a threshold that is larger than the drift. Weinberg's step length estimation approach relies solely on the difference between a_{max} and a_{min}, rendering the actual value inconsequential.

Using PDR for localization introduces several constraints on the user, which must be adhered to for accurate positioning. Specifically, the smartphone must be held at the front of the body in an upright position. The user must minimize arm-swinging during walking to prevent erroneous position estimates. Moreover, the phone's camera must always face the direction of walking to ensure that step updates are applied in the correct direction. These constraints make the localization process less intuitive and natural, increasing the burden on the user.

Additionally, individual variations in walking patterns can considerably affect the accuracy of PDR-based localization. An individual's walking pattern includes the degree of bouncing during walking, variability in step size, and the speed of turning. For example, abrupt turns around the user's own axis, which are sensed not only by the gyroscope but also by the accelerometer, can be mistakenly interpreted as steps taken. These false positives in the step detection may result in incorrect position estimates, particularly if such scenarios occur frequently.

The results of PDR are presented in Figure 5.7. The final position estimate demonstrates a high level of accuracy. In this experiment, the step length estimation was performed using a fixed

value of 70 cm. The orientation of the calculated path mostly aligns correctly with the actual path taken. However, it should be noted that the very first step appears to have been missed in some of the trials, as indicated by the slight deviation at the curve of the path. This behavior is attributed to the sensitivity of the step detection algorithm, which may occasionally fail to detect steps, particularly at the beginning and end of walking periods. It is important to note that the path was traversed by a user who carefully followed the prescribed walking constraints. Users with more relaxed or irregular walking patterns may achieve less accurate results.

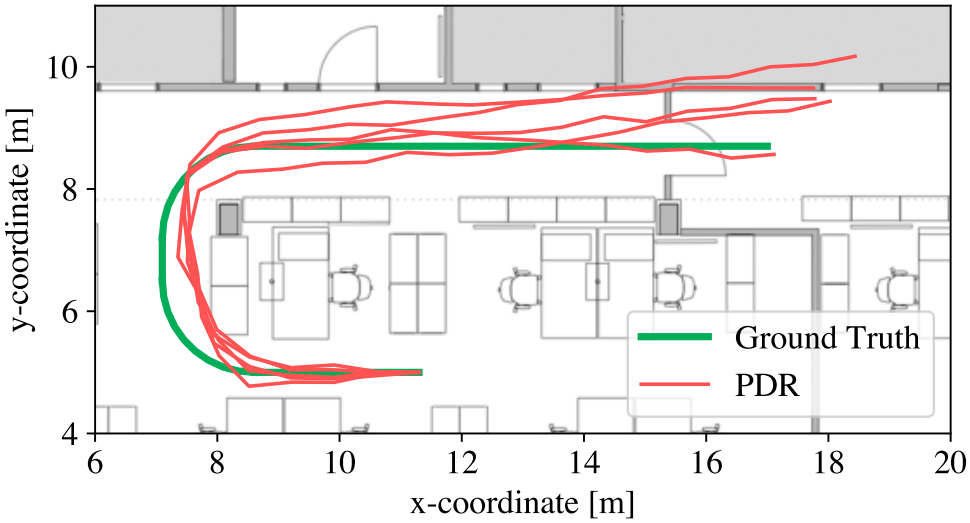


Figure 5.7.: *Results of* PDR *localization*

Step Length Estimation

We proposed two distinct techniques for estimating step length during the PDR localization process. Weinberg's method is a dynamic approach that relies on the difference between the maximum and minimum acceleration values obtained during a step. In contrast, the static approach uses a predetermined step length for all steps taken during the localization process.

In our comparative evaluation of the two step length estimation methods, we observed that the static approach produced more consistent and reliable outcomes. Our experiments revealed that utilizing Weinberg's method often resulted in overestimation of step lengths, as shown in Figure 5.8. This outcome may be attributed to the original design of Weinberg's method for use in pedometers. The effect could be minimized by introducing a scaling factor. Although the dynamic approach may perform well for experienced users, the numerous constraints imposed by it can be overly complex for novice users. Holding the smartphone in a stable position and avoiding arm movements during walking may not come naturally to them. As a consequence, the static approach is more appropriate for a broad range of users seeking a straightforward and accessible localization technique. It is less susceptible to imperfect posture because the acceleration data are utilized solely for step detection and does not affect the step length.

The results obtained from the static approach demonstrate high accuracy. Nevertheless, it is crucial to acknowledge that the experiments showcased in Figure 5.9 were carried out by an

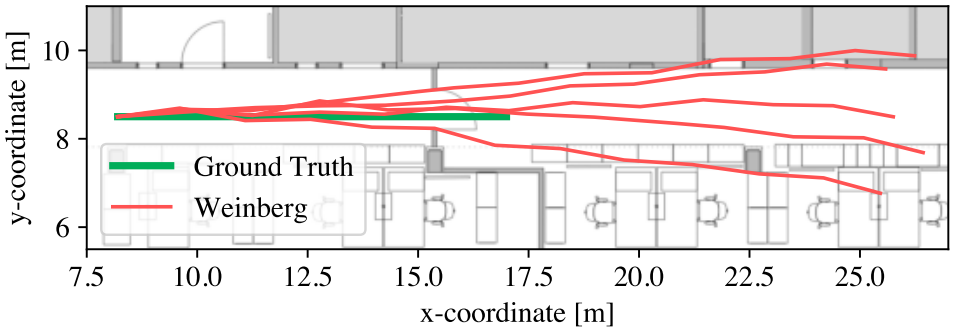


Figure 5.8.: *Results of* PDR *with Weinberg's step length estimation*

experienced user under controlled conditions. The user maintained a consistent walking speed, leading to relatively constant step lengths. In real museum settings, however, users tend to exhibit more diverse step lengths as they navigate through different areas. Factors such as walking short distances between nearby artworks or traversing large halls at a faster pace contribute to variations in step length. These variations impact the accuracy of step length estimation, particularly when employing a fixed value across all scenarios.

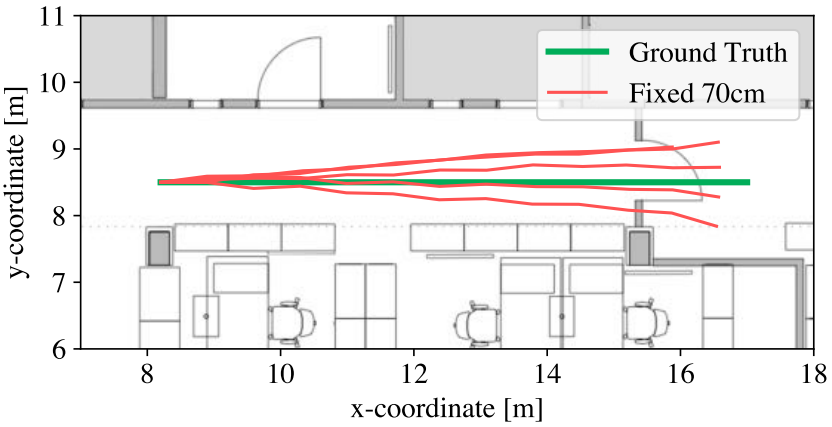


Figure 5.9.: *Results of* PDR *with a fixed step length of* 70 *cm*

5.1.4. Discussion

In summary, the PDR approach yields superior results compared to dead reckoning in indoor localization. This result is due to the extremely inaccurate sensor data, which lead to increasingly large errors in the dead reckoning approach. We do not advise direct use of sensor data, as they exhibit large deficiencies. The dead reckoning approach is highly susceptible to inaccuracies stemming from noisy sensor data, and drift accumulation poses an important challenge to eliminate. Even with the application of several methods such as ZUPT and calibration, dead reckoning remains an unreliable means of localizing a person indoors. As shown before, dead reckoning is even unable to correctly estimate movement along a straight line.

While the PDR approach does not suffer from the same issues as dead reckoning, it has its own

set of challenges. If the user correctly holds the phone and the step length is adjusted to the individual, PDR yields quite accurate results. However, for the vast majority of users, results are mixed due to their more relaxed behavior of not rigidly following the constraints.

In general, sensor-based localization poses a big challenge in achieving accurate results due to the presence of noise and drift in the sensors. The noise renders the data unpredictable, and the results may vary depending on the specific smartphone model. This variability presents a further challenge to an application that is designed be used on a wide range of devices.

5.2. User Study

We conducted a user study to investigate the effectiveness of different methods for displaying navigation information in AR. The goal of the study was to identify differences among the methods and to provide design recommendations for future AR applications.

5.2.1. Design

In this section, we present the design of our user study which aimed to compare the three methods *fixed*, *corner*, and *many*, which were described in Section 4.5. An overview of the study design is provided in Figure 5.10.

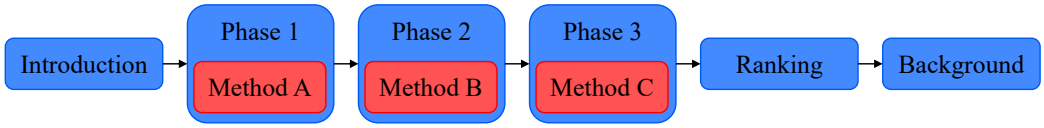


Figure 5.10.: *Study design*

Upon arrival at the study session, participants were provided with an introduction about the goal of the study and a demonstration of how the app works. The study took place in the GTC office space, where a small exhibition was set up. As part of the introduction, we obtained an estimate of each participant's step length by having them walk a distance of 10 meters while counting the number of steps taken. This information was collected to improve the localization accuracy in the study and used as a fixed number instead of the dynamic step length estimation.

The main part of the study consisted of three phases, each with the same structure. In each phase, participants followed a predefined path while being guided by one of the three methods *fixed*, *many*, or *corner*. For each phase, participants filled out a questionnaire on the method used after completing the path. They then proceeded to the next phase with a different method. After completing all three phases, participants ranked the three methods and provide additional information about their personal background. We asked about personal background at the end of the study to minimize bias. This design ensured that participants' responses to the study's main content were not influenced by their awareness of being categorized or studied based on their demographic characteristics.

The order of the methods was randomized among the participants to eliminate bias. For example, while the first participant may have had the *fixed* method in phase 1, the second participant may have had the *many* method in that phase. However, the path for each phase remained fixed, ensuring that all methods were tested under the same circumstances. This design minimized the impact of different paths on the results and allowed for a direct comparison between the methods in a controlled manner.

All artworks used in the study were embellished with a golden frame. This frame served the purpose of distinguishing artworks of interest from other unrelated artworks in the office. The frame was obtained from Freepik[1], a website that offers various digital assets.

Path per Phase

Each phase of the study consists of a path composed of three sub-paths, as illustrated in Figure 5.11. Participants began each phase by tracking an artwork, which served as the starting point, and then sequentially followed the guidance to three other artworks. Each path included an artwork where participants were required to turn around 180° to continue, and one sub-path that was a straight walk along a corridor. This design was implemented to ensure comparability between the phases. Care was taken to arrange the paths in such a way that participants only saw the artworks when they were guided to them, thus minimizing the likelihood that they would know where to go without following the instructions. As a result of these constraints, the lengths of the paths differed, with phase 1, phase 2, and phase 3 approximately measuring 30, 27, and 19 meters, respectively. However, since the order of the methods was randomly assigned among participants, the path length did not influence the results for each method.

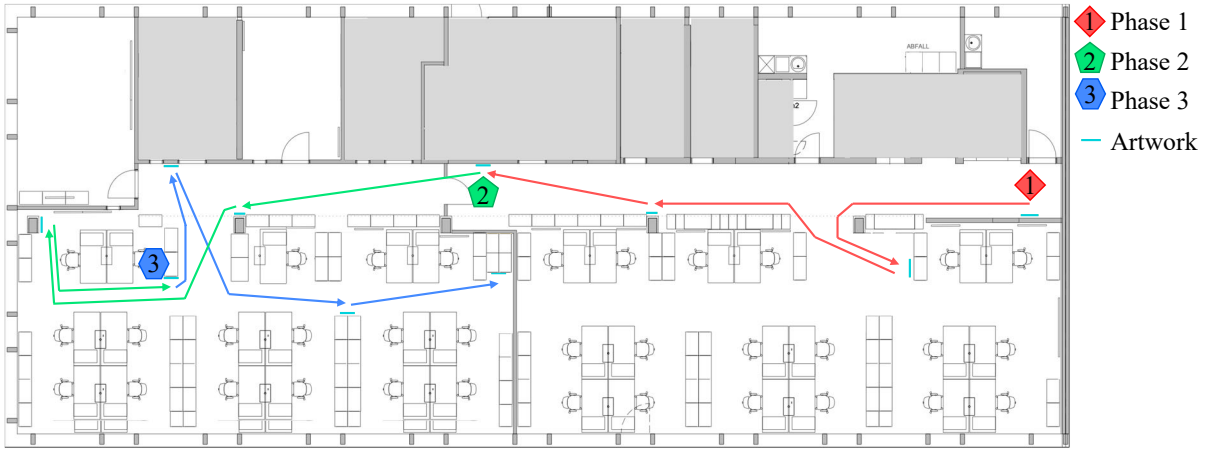


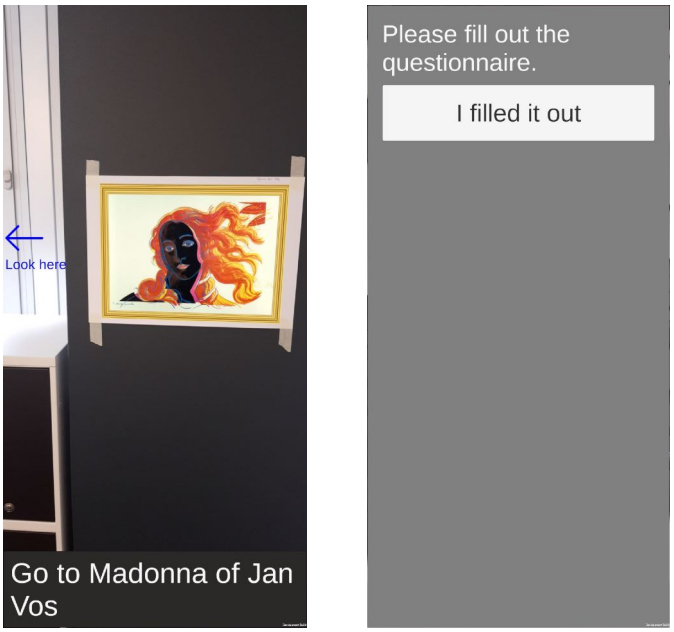
Figure 5.11.: *The three paths of the user study with sub-paths*

Changes made to App

For the user study, we changed some designs and functionalities of the app. To focus solely on the three methods for displaying the path in AR, the map and the ability to switch between

[1] `https://www.freepik.com/` (Accessed: 24. May 2023)

views were removed from the UI during the study. This change eliminated any potential differences among participants due to their ability to read maps. The simplified UI, as shown in Figure 5.12 (a), displayed instructions at the bottom, including the name of the next artwork. These instructions were the only navigation information provided to the user during the study apart from the more detailed instructions via one of the three methods. Users were automatically guided through the three phases with instruction screens in between telling them to fill out the questionnaire, as shown in Figure 5.12 (b). The paths were hard-coded to minimize the need for interaction during the study, so no input was required from the users.



(a) Study UI **(b)** Instruction screen

Figure 5.12.: *Simplified user interface for user study*

To minimize the influence of localization errors and focus on the method of displaying navigation information, we used additional mechanisms for the study. One such mechanism was the constraint that steps were not taken in the direction of the phone, but rather along the calculated path towards the next artwork. This constraint eliminated potential errors in localization when the user did not hold the phone in a way that is precisely in alignment with their direction of travel. Users could move more freely in the way they preferred, and differences in localization accuracy between participants were reduced. This constraint helped ensure that any observed differences in user preferences among the methods were primarily due to the method of displaying navigation information, rather than variations in localization accuracy. We called this constraint ”aiding” on the settings page as it aided to update the position in the correct direction.

To enhance localization accuracy and reduce potential errors, we implemented a mechanism that allows the study instructor to correct the estimated position of the user. We developed a web app that allows the instructor to monitor the estimated position of the participant and compare it with their actual position. If there is a significant discrepancy, the instructor could reset the position by clicking on a point on the floor plan, thus correcting the localization error. We refer to this process of updating the current position estimate with a more accurate one as

a ”correction”. This mechanism helped ensure that the data collected during the study were as accurate as possible.

To ensure consistency and accuracy in measuring participants' movements, we opted for a fixed step length for each participant throughout their study session, as opposed to estimating step length based on acceleration data. During testing prior to the study, we observed large differences in step length estimates due to variations in users' walking styles, such as bouncing or minimal vertical movements. Therefore, we use a fixed step length throughout the study. This approach ensured that any differences observed are more likely to be attributed to the effectiveness of the method to display navigation information, rather than individual differences in step length or walking style.

Questionnaire

The questionnaire used in this study comprises four parts. The first three parts are identical and are completed after each phase of the study. Each of these parts includes the standardized System Usability Scale (SUS) questionnaire [Bro95] to obtain a usability score for the respective method. The SUS score can be interpreted using different scales, as illustrated in Figure 5.13 [BKM09]. The adjective rating scale in particular provides insight into the interpretation of the resulting scores.

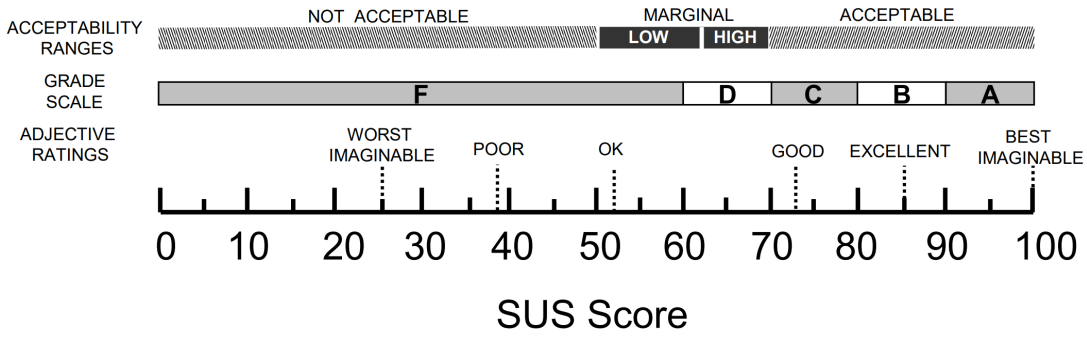


Figure 5.13.: *Different scales / ratings in relation to* SUS *scores*

The questionnaire also contains of a series of questions designed to gather detailed feedback on each method to display directions. To reduce bias and capture nuances in participants' experiences, each topic was addressed with both positive and negative formulations. The questions are presented below, ordered in a way that highlights these pairs of questions belonging to the same topic. Participants were asked to rate their responses on a scale of 1, strongly disagree, to 7, strongly agree. In the study, the questions were presented in a slightly different order than shown here. The order of questions used in the user study is shown in Section A.2.

- I knew where I had to go.
- I felt lost.
- The directions were very clear.
- The directions were given in a confusing way.

- It was easy to follow the path.
- Getting to the artworks was challenging.
- The app controlled my movements.
- I felt in control of my movements.
- It was an intuitive way to display directions.
- The directions felt unnatural to me.
- I can imagine using this application at a museum.

The fourth and final part of the questionnaire includes ranking questions for the three methods, as well as questions related to the participant's personal background. The ranking questions were designed to assess preferences among the three methods. The background questions focused on demographic information, previous experience with AR applications, and familiarity with office space, as these factors can influence the study results. The specific background questions used in the study are presented in Section A.2. The ranking questions are shown below in the same order as used in the user study. Participants were asked to assign a unique rank from 1 to 3 to each method. Rank 1 corresponds to the method that participants perceived as best fulfilling the question among the three methods. This ranking system allows for a direct comparison of the methods in terms of the participants' preferences and perceptions.

- Which method felt most natural to you?
- Which method gave the clearest navigation instructions?
- Which method did you find the most visually appealing?
- Which method would you be most likely to use at a museum?
- Overall, which method did you like the best?

Logged Data

For each participant, we collect data on the time taken to traverse each path and the number of corrections that were made during the navigation process. These variables potentially influence the perceived usability of the method to display directions.

Statistics

To assess the results of the questionnaire, a statistical test is carried out to determine the significance of the differences between the three methods. Accordingly, we formulate the null hypothesis $\mathcal{H}_0$ and the alternative hypothesis $\mathcal{H}_A$:

$\mathcal{H}_0$: There are no significant differences among the results of the three methods

$\mathcal{H}_A$: At least one of the methods exhibits a significant difference from the others

The test that is best suited for this purpose is the one-way repeated measures Analysis of Variance (ANOVA) test. This test has three prerequisites: a normally distributed response variable, homogeneous variance among the methods, and independence of observations. The normality assumption is checked using the Shapiro-Wilk normality test [SW65]. Homogeneity of variance is checked with Levene's test [Lev61]. The independence of observations is guaranteed by the study design, which randomizes the order of the methods.

If the data fail to meet all requirements, Friedman's ANOVA test is used, as it does not make any assumptions on the distribution of the underlying data. It is a non-parametric test that ranks the data and uses these ranks instead of the actual values to perform the test, which makes it less sensitive to extreme values and non-normality. In case of significant differences, we apply a pairwise Wilcox test to determine between which groups the differences lie. If the significance level, denoted as "p-value", is less than 0.05, we reject $\mathcal{H}_0$ and accept $\mathcal{H}_A$. This value is commonly used as a threshold for statistical significance.

To compare the results obtained from the questionnaire with the metrics collected during the study, we use statistical methods such as Pearson's correlation coefficient [CHC+09] or Spearman's rank correlation coefficient [Spe08], depending on the normality of the data. Pearson's correlation coefficient is used when the data follow a normal distribution and there is a linear relationship between the variables being compared. On the other hand, Spearman's rank correlation coefficient is a non-parametric measure that, similar to Friedman's ANOVA, uses the rank of the data rather than the actual values. The resulting correlation coefficient ρ is between -1 and 1 and indicates the strength of the correlation. A value close to zero corresponds to a weak correlation.

5.2.2. Results

In this section, we present the results of the user study conducted to evaluate the three methods for displaying navigation information. We conducted the Shapiro-Wilk test for normality on the response data. The results indicated that none of the responses followed a normal distribution. The non-normality in the data means that the assumptions for one-way repeated measures ANOVA are not met. Therefore, we applied Friedman's ANOVA to check for significant differences in all results. The statistical analysis was performed using the R software package.

In cases where the p-value is reported as less than 0.01 or greater than 0.1, we want to indicate that the results are particularly clear or obvious.

Details

A total of 30 participants, consisting of 6 women and 24 men, were recruited for the user study. The age of the participants ranged from 22 to 35 years, with a mean age of 27 years. Half of the participants were familiar with the study environment prior to the study, while 13 participants were completely unfamiliar, and 2 had some prior exposure to the environment. Only one participant had no previous experience with AR applications, while 4 participants used AR applications daily, 8 used them weekly, and 6 used them monthly. Furthermore, 11 participants reported using AR applications less than once a month.

The mean step length in the study was 68 cm, with a minimum of 59 cm and a maximum of 77 cm. We observed noticeable variations in the walking styles of participants during the study, with some individuals exhibiting very pronounced bouncing movements while others walked cautiously with minimal vertical motion. These differences in walking styles may have influenced the accuracy of the localization, as the amount of motion of the phone affects the step detection. Participants with more pronounced bouncing movements may have experienced a higher localization accuracy due to more accurate step detection compared to participants who walked more steadily with minimal vertical movement. The sharpness with which participants turned corners during the navigation was also found to be a factor that influenced localization accuracy. Specifically, quick turns made by participants were sometimes interpreted as steps or movements by the localization system, leading to inaccuracies in estimating the user's position and orientation.

System Usability Scale

The results of the SUS analysis indicate that there are no statistically significant differences between the three methods ($p > 0.1$). However, we did observe a significant difference between the three phases of the study ($p < 0.01$). The results of a post-hoc Wilcox test indicate that the second phase, which featured a more complex path, was less enjoyable for the users. It is noteworthy that while the complexity of the path appears to have influenced the user experience, it does not impact the performance of any particular method. The order of the methods was randomized to ensure that all methods were tested under similar conditions.

The SUS scores for each method are presented in Figure 5.14 using a violin plot, which shows the distribution of the data points. The plot includes a box in the middle representing the interquartile range, which is the middle 50% of the data, and a white dot representing the median value. The width of the violin indicates the density of the data points, while the thin lines extending from the box indicate the data range excluding outliers. The length of the violin represents the range of the data including outliers.

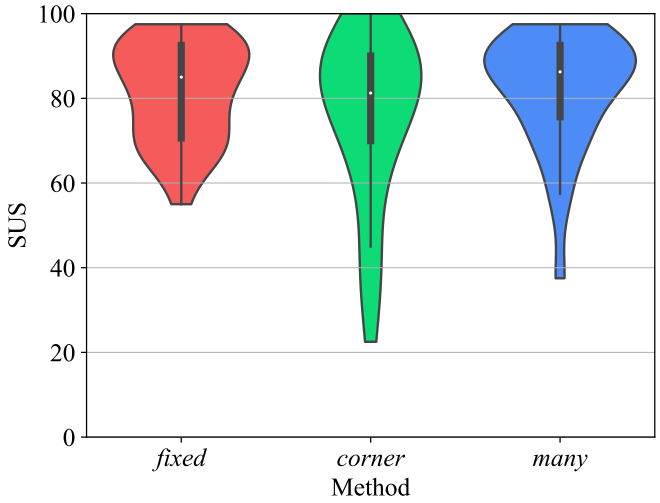


Figure 5.14.: *Violin plots of* SUS *score per method*

The analysis of the obtained results reveals that the *corner* method exhibits a slightly lower

median SUS score compared to the other two methods, albeit having a larger range of values. In contrast, for the *fixed* and *many* methods, the plot presents the highest density of data points at a high SUS score. On the basis of these plots, it can be confirmed that the difference in the SUS scores among the methods is not statistically significant. Further supporting evidence is provided by the adjective scale analysis introduced in Figure 5.13, which indicates that all methods are rated between good and excellent. Specifically, the average SUS scores are 81.5 for *fixed*, 75.9 for *corner* and 82 for *many*.

Other Questions

Table 5.1 presents the means of the answers to the remaining questions in the study. It is worth noting that some participants expressed confusion about questions Q6 and Q11, which addressed the feeling of control, rendering the results of these questions less meaningful than others.

For questions Q1-Q6, a high score is preferable, while for Q7-Q11, a low score is desired. The table highlights that the *many* method scored best in almost all questions. Despite the clear trend, the differences between the three methods are not statistically significant ($p > 0.1$ for all questions).

Nr	Question	*fixed*	*corner*	*many*
Q1	I can imagine using this application at a museum.	5.47	5.13	5.77
Q2	I knew where I had to go.	5.50	4.97	5.67
Q3	It was easy to follow the path.	5.37	4.77	5.70
Q4	The directions were very clear.	4.93	4.50	5.40
Q5	It was an intuitive way to display directions.	5.70	5.17	5.67
Q6	I felt in control of my movements.	4.80	5.47	5.40
Q7	The directions were given in a confusing way.	2.67	3.37	2.47
Q8	Getting to the artworks was challenging.	2.17	2.67	2.17
Q9	I felt lost.	2.40	2.83	2.23
Q10	The directions felt unnatural to me.	2.77	2.60	2.57
Q11	The app controlled my movements.	5.43	4.43	5.07

Table 5.1.: *Average answers per method and question*

Rankings

The average rank of each question is presented in Table 5.2. It is evident that the *many* method received the best ranking on all the questions except for the one related to visual appeal, where it was ranked the worst (a low rank means that the method is preferred to others).

Nr	Question	*fixed*	*corner*	*many*
Q1	Which method felt most natural to you?	2.03	2.17	1.80
Q2	Which method gave the clearest navigation instructions?	2.03	2.43	1.53
Q3	Which method did you find the most visually appealing?	1.77	2.00	2.23
Q4	Which method would you most likely use at a museum?	2.13	2.23	1.63
Q5	Overall, which method did you like best?	2.03	2.20	1.77

Table 5.2.: *Average rank per method*

Significant differences were observed in the rankings of Q2 and Q4. The p-values of pairwise comparisons based on the Wilcox test are presented in Table 5.3. Regarding Q2 about the method that provided the clearest instructions, we found significant differences between *many* and *fixed* ($p < 0.05$), as well as between *many* and *corner* ($p < 0.0005$), but not between *fixed* and *corner* ($p > 0.1$). The *many* method was ranked highest in terms of providing the clearest instructions. In particular, the difference between *many* and *corner* was much greater than the difference between *many* and *fixed*, as evidenced by the respective p-values (i.e. lower p-values indicate stronger differences). Furthermore, the *many* method was reported to be significantly more likely to be used at a museum than the *corner* method ($p < 0.05$), but no other pairs showed significant differences.

	fixed	***corner***
corner	0.12	-
many	0.034	0.00013

(a) Clearest instructions

	fixed	***corner***
corner	1.00	-
many	0.058	0.014

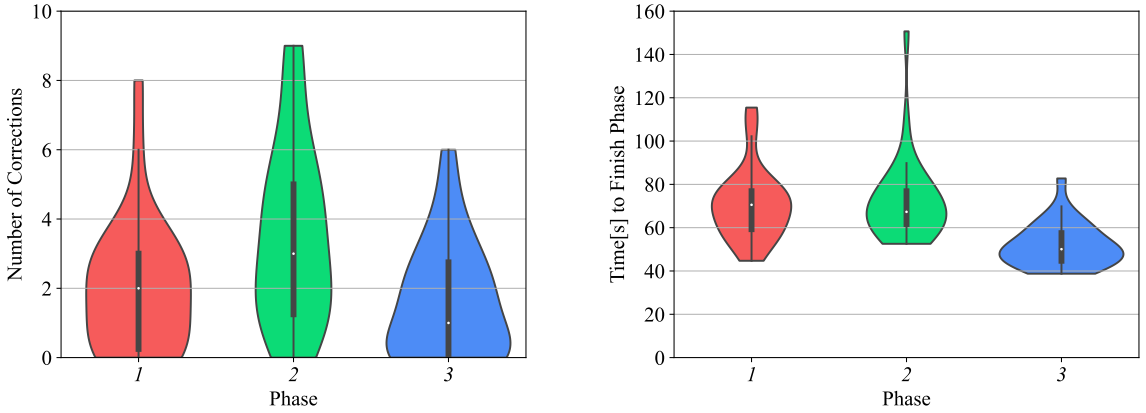
(b) Preferred at a museum

Table 5.3.: *p-values for pairwise comparisons*

Logged Data

The analysis of the data revealed that there was no statistically significant difference in the number of corrections made during the session among the three methods. However, a significant difference was found among the three phases ($p < 0.01$), with phase 2 showing significantly more corrections compared to the other phases. This difference is clearly visible in Figure 5.15 (a), as the violin of phase 2 is thinner and larger than those of the other phases. This observation is consistent with the higher complexity of the second path, as mentioned in the results of the SUS scores.

Similarly, when examining the time it took participants to complete the full path, no significant differences were observed between the methods. However, a significant difference was found between phases ($p < 10^{-6}$), with phase 3 requiring a significantly shorter amount of time compared to the other phases. Again, this effect is clearly visible in Figure 5.15 (b), this time

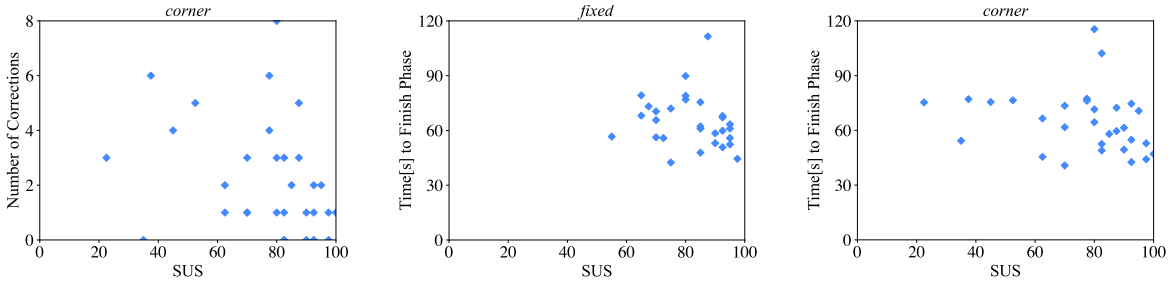


(a) Number of corrections **(b)** Time to finish

Figure 5.15.: *Results of logged data*

due to the thicker and shorter body of the violin. This result was expected as the path of phase 3 was shorter than the paths of the other two phases.

Since the SUS scores do not follow a normal distribution, we use Spearman's rank correlation coefficient to check whether the SUS score is correlated with any of the metrics collected in the study. A significant negative correlation was identified between the number of corrections made and the SUS score obtained for the *corner* method, with a statistically significant $p < 0.05$ and a moderately strong correlation coefficient of $\rho = -0.39$. This coefficient indicates that a higher number of corrections was associated with lower SUS scores for that particular method. However, no significant correlations were observed for the other methods. The associated data points are shown in Figure 5.16 (a). Negative correlation can be perceived as a trend line that approximates the data points with a negative slope.



(a) SUS and number of corrections for *corner* **(b)** SUS and time to finish for *fixed* **(c)** SUS and time to finish for *corner*

Figure 5.16.: *Correlations of logged data with* SUS *scores*

When comparing the time taken to complete the path to the SUS scores, negative correlations were found for both the *fixed* and *corner* methods ($p < 0.05$ and $\rho = -0.40$), indicating that a longer time was associated with lower reported SUS scores for these methods. The plots are shown in Figure 5.16 (b) and (c). For the plots of uncorrelated data points, refer to Section A.4.

Informal Feedback

Table 5.4 presents an overview of the feedback provided for each method. The primary advantage of the *fixed* method was to consistently display information in front of the user, eliminating the need to search for the arrow in the surroundings. The design of the method was also considered clear in terms of providing guidance at each point in time. However, it was observed that this method required constant attention on the screen, resulting in reduced awareness of the surroundings and limited ability to view nearby artworks during the navigation. Additionally, four participants mentioned that they would have preferred more advanced information about the upcoming path, as the method only provides instructions for the immediate moment without indicating the next steps.

	Advantages	**Disadvantages**
fixed	- always there, no searching - clear what to do at each point in time	- need to watch screen - no information on what is ahead
corner	- know ahead what to do - simple instructions	- hard to find when far away, small - inaccuracies very noticeable
many	- view path ahead, very clear - walk independently	- arrows too close & too large - overwhelming instructions

Table 5.4.: *Feedback given on each method to display directions*

The *corner* method was noted to provide clear information about the location of the next corner and the direction to turn, allowing users to anticipate the upcoming steps. This feature was appreciated by the participants for its forward-looking guidance. The instructions were perceived as simple and clear. However, five participants reported difficulty in locating the small arrow on the screen when the next corner was distant. Additionally, the *corner* method was found to be more sensitive to localization inaccuracies, as the single symbol provided may lead to confusion if it is not precisely aligned with the actual corner location. On the contrary, the *many* method was found to be more forgiving in accommodating small localization errors due to the availability of multiple arrows for reference.

The *many* method provided clear instructions since it allowed participants to view a substantial portion of the path ahead. This view enabled them to navigate to the next corner independently without constantly referring to the screen. The ability to also know the direction of the next turn at each corner was appreciated by the participants. However, 16 users expressed concerns about the density of information on the screen, with comments indicating that the arrows were too close together and appeared too large, leading to a sense of overwhelming instructions.

In addition to feedback specific to the three methods, the participants provided comments on general features of the application. For example, the hint provided on which direction to turn to view the arrows was perceived as a highly useful feature. Six participants mentioned that they liked this particular feature. It prevented them from aimlessly searching for the arrows, enhancing their navigation experience.

Three participants expressed appreciation for any feedback from the app that confirmed they were on the right track. Specifically, they liked the overlay of the image over the tracked artwork, which indicated successful tracking. The change of UI when they got close to the artwork was also well-received as it provided a sense of accomplishment and success.

Another issue reported by six participants was the lack of clear information about the end point of the path. Instructions were designed to include only a single symbol (the arrow). The participants expressed that this design led to confusion, as they were unsure whether an arrow was indicating a turn or pointing to the destination. To address this issue, they suggested having a different symbol for the artwork itself, which would allow them to identify the destination from a distance. This improvement would resolve the issue of ambiguity when the artwork was close to a corner.

A comment one participant made was about the direction of the path. The current design calculates the path as the shortest line to the destination, which this participant found confusing. They expected the path to be in the middle of the corridor, rather than a straight line along the corridor when the destination is further away on the other side of the corridor. This situation is visualized in Figure 5.17. The issue arises as a result of utilizing the NavMesh for pathfinding. Utilizing a waypoint graph could resolve this problem. It should be noted that even with the design requested by the participant, there may still be inaccuracies in localization that result in the path appearing crooked.

(a) Our design

(b) Design requested by the participant

Figure 5.17.: *Path along corridor*

5.2.3. Discussion

The findings of this study reveal a clear trend towards the *many* method based on the rankings, with the only exception being the visual appeal. Further analysis of the informal feedback indicated that the overwhelming feeling of having too many arrows on the screen contributed to this exception. The optimal method for displaying navigation information would likely be a hybrid approach that combines the advantages of both the *many* and *corner* methods, providing clear instructions on the path ahead while avoiding excessive redundancy on the screen. This hybrid approach would also address the issue of the arrow being excessively small when it is located far away in the *corner* method.

When comparing the three phases of the study, it was observed that phase 2 had a higher number of corrections and a lower average SUS score. This difference may be attributed to the increased complexity of the path in phase 2, which involved sharp turns in an environment with numerous possibilities for turning. As a result, users had to rely more heavily on the localization, and any inaccuracies in the system were perceived as more problematic and made for a less pleasant

experience. In contrast, in environments with fewer ambiguities, localization inaccuracies were less detrimental, and users were able to recover from them more easily. While this observation does not impact the comparison of the three methods, it highlights the importance of providing reliable assistance in complex environments where users may require additional support to navigate effectively.

Our findings indicate negative correlations between the SUS score and the number of corrections made for the *corner* method, as well as between the SUS score and the time taken to traverse the path for the methods *corner* and *fixed*. One possible explanation for the lack of correlations in the *many* method could be that localization inaccuracies have less impact on this method. In the *many* method, it is easier to understand the general direction of the path, even if the orientation or distances of the shown instructions are slightly off. However, in the *fixed* and especially in the *corner* method, such inaccuracies are confusing and lead to difficulties in identifying the correct path. Consequently, when the localization accuracy is low, users may experience more confusion and find the method less enjoyable. Low localization accuracy often results in more corrections, which in turn require additional processing time by the user, leading to longer traversal times to reach the destination. The number of corrections made can therefore serve as an indicator of the localization accuracy experienced by each participant.

As previously stated, the walking patterns, phone stability, and sharpness of turns exhibited by the participants showed considerable variability. These individual differences in walking styles highlight the necessity of considering user behavior and movement patterns when evaluating localization systems in real-world settings, as they can influence their accuracy. Additionally, these findings suggest that localization systems that are less reliant on individual user behavior may be more suitable for environments where a diverse range of individuals utilize the system, which is the case for a museum app.

The informal feedback obtained from the users consistently indicated that increased feedback from the app enhanced their confidence in using the system and contributed to a more enjoyable user experience. Therefore, we recommend features that provide users with information about the status of the system, such as displaying a map with the estimated position to aid in orientation. We also recommend giving visual feedback to indicate to users that they are correctly tracking an artwork.

The implementation of the path along a corridor in the current system is not parallel to the corridor itself, as dictated by Unity's NavMesh. However, this design choice has limitations. It may not align with the intuitive expectations of smartphone users, who are accustomed to seeing paths displayed in alignment with the real-world environment. As such, solutions that aim for higher acceptance among users should consider incorporating straight paths in their design. This design aligns better with users' expectations and provides a more intuitive navigation experience.

6

Conclusion

This thesis aimed at developing an AR application for indoor navigation within the museum context. For the purpose of localization, our approach exclusively utilizes the sensors that are readily available on the device, namely the accelerometer, the gyroscope, and the camera. The developed application has two main components. The first is the floor plan editor that enables the user to set the artworks' positions and define the walkable areas of the museum. The second component is an application that guides the user to a previously selected artwork destination using AR instructions.

When the user is tracking an artwork, the system calculates the user's position based on the tracking information. When the user is not tracking any artworks, the application utilizes one of two available methods to update the user's position, namely dead reckoning and PDR. Dead reckoning updates the position by integrating the acceleration and the angular velocity, which leads to a substantial accumulation of errors. Neither the calibration of the acceleration nor the ZUPT method have effectively eliminated the errors. PDR employs step detection, step length estimation, and heading estimation to update the user's position, producing more accurate results with less sensitivity to noise. However, this method requires the user to walk and hold their phone in a specific way that may feel unnatural to inexperienced users.

We conducted a user study to evaluate the three different methods of displaying directions in AR. The results suggest that users prefer to know the route in advance to avoid staring at the screen while navigating. However, the proposed *many* method was too visually overwhelming. The optimal solution would be an approach that combines elements from both the *many* and the *corner* method, striking a balance between clarity of instructions and a simple, unobtrusive display.

6.1. Limitations and Future Work

One of the current system's limitations is that it is constrained to a single floor, thereby offering a limited scope for application in larger settings. The floor plan editor demonstrates potential for enhancements, for example incorporating multiple floors and supplementary amenities, such as restrooms and elevators. Furthermore, the current system does not allow for adding new floor plans or artworks. Adding such a feature would improve the floor plan editor's flexibility.

Localization using sensor data is inherently affected by noise and variations between individuals, which introduce errors and diminish the reliability of the position estimate. Although PDR can provide precise position estimates in optimal conditions, its accuracy substantially deteriorates when users fail to comply with the prescribed constraints regarding phone handling. Our experiments clearly show the limitations of both PDR and dead reckoning. Consequently, we recommend exploring alternative localization techniques, such as lateration or fingerprinting, which may offer better accuracy and robustness under varying circumstances. Alternatively, there is potential in utilizing vision-based methods, particularly when integrated with floor plan information.

A

Appendix

This chapter presents additional information for those interested.

A.1. Zero Velocity Potential Update Thresholds

Table A.1 shows the thresholds we used to detect ZUPT conditions.

rot_x	rot_y	rot_z	$\|acc\|$
$\pm 4°$	$\pm 3°$	$\pm 4°$	$\pm 0.4 m/s^2$

Table A.1.: *Thresholds to detect standing still*

A.2. User Study Questionnaire

Below are the questions we asked in each phase of the user study, in the order they were presented after the SUS questions.

To what extent do you agree with the following statements? (1 = strongly disagree, 7 = strongly agree)

- I knew where I had to go.
- The directions were given in a confusing way.
- It was easy to follow the path.

- The app controlled my movements.
- The directions were very clear.
- I felt lost.
- I felt in control of my movements.
- The directions felt unnatural to me.
- It was an intuitive way to display directions.
- Getting to the artworks was challenging.
- I can imagine using this application at a museum.

The following questions have answer options available or are to be answered either in writing or in talking to the study coordinator.

- Did you experience any problems? - (none, minor one(s), severe one(s))
- What did you particularly like about this method?
- What did you find most challenging about this method?

After finishing the three phases, study participants filled out the ranking questions as presented in Section 5.2.1. The final part of the study consisted of the following questions, asking about background and demographics.

- Were you already familiar with the office space? - (yes, somewhat familiar, no)
- Did you know any of the artworks beforehand? - (none, one, two, more than two)
- How often do you use AR applications? (including e.g. Snapchat filters, Google Live View, ...) - (daily, weekly, monthly, less than monthly, This is my first time using an AR application)
- Gender - (female, make, other / prefer not to say)
- Age
- Profession

A.3. Friedman's Analysis of Variance

Table A.2 presents the p-values obtained when applying Friedman's ANOVA test on the data. For questions Q1-Q13, the test was conducted to compare the responses obtained from the three methods, namely *fixed*, *many*, and *corner*. For questions Q14 and Q15, the data were compared across the three phases, which translates into comparing the paths that the participants took for each method. Questions Q16-Q20 were tested by considering the ranks given to each of the three methods.

The green cells indicate statistically significant results. It is noteworthy to mention that the data indicate a significant difference among the three methods for Q11. However, this finding was

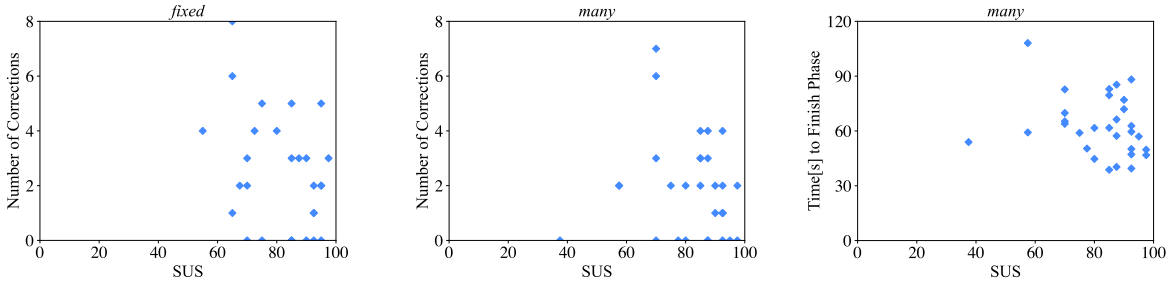
Nr	Question	p-value
Q1	I can imagine using this application at a museum.	0.16
Q2	I knew where I had to go.	0.38
Q3	It was easy to follow the path.	0.095
Q4	The directions were very clear.	0.10
Q5	It was an intuitive way to display directions.	0.22
Q6	I felt in control of my movements.	0.10
Q7	The directions were given in a confusing way.	0.12
Q8	Getting to the artworks was challenging.	0.52
Q9	I felt lost.	0.62
Q10	The directions felt unnatural to me.	0.80
Q11	The app controlled my movements.	0.037
Q12	SUS for each method	0.72
Q13	Number of corrections for each method	0.32
Q14	SUS for each phase	0.0069
Q15	Number of corrections for each phase	0.0010
Q16	Which method felt most natural to you?	0.36
Q17	Which method gave the clearest navigation instructions?	0.0022
Q18	Which method did you find the most visually appealing?	0.20
Q19	Which method would you most likely use at a museum?	0.045
Q20	Overall, which method did you like best?	0.24

Table A.2.: *Results of Friedman's* ANOVA *per question*

not further discussed in the results section due to the post-hoc Wilcox test failing to identify any pairs with a significant difference. Additionally, considering that Q11 was one of the questions that frequently confused the participants, this result was deemed inconsequential and was not included in the analysis.

A.4. Correlation Plots

Figure A.1 shows the plots for the data points that do not exhibit a significant correlation.



(a) SUS and number of corrections for *fixed*

(b) SUS and number of corrections for *many*

(c) SUS and time to finish for *many*

Figure A.1.: *Correlations with* SUS *scores*

Bibliography

[BKM09] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *J. Usability Studies*, 4(3):114–123, may 2009.

[Bro95] John Brooke. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, 11 1995.

[CHC+09] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4, 2009.

[DWZ12] Buti Al Delail, Luis Weruaga, and M. Jamal Zemerly. Caviar: Context aware visual indoor augmented reality for a university campus. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 286–290, 2012.

[DWZN13] Buti Al Delail, Luis Weruaga, M. Jamal Zemerly, and Jason W. P. Ng. Indoor localization and navigation using smartphones augmented reality and inertial tracking. In *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*, pages 929–932, 2013.

[Ebe08] David Eberly. Triangulation by ear clipping. *Geometric Tools*, pages 2002–2005, 2008.

[Haq14] Israat Tanzeena Haque. A sensor based indoor localization through fingerprinting. *Journal of Network and Computer Applications*, 44:220–229, 2014.

[HNR68] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[HPS+14] Hui-Huang Hsu, Wei-Jan Peng, Timothy K. Shih, Tun-Wen Pai, and Ka Lok Man. Smartphone indoor localization with accelerometer and gyroscope. In *2014 17th International Conference on Network-Based Information Systems*, pages 465–469, 2014.

[HTJ16] Ngoc-Huynh Ho, Phuc Huu Truong, and Gu-Min Jeong. Step-detection and adaptive step-length estimation for pedestrian dead-reckoning at various walking speeds using a smartphone. *Sensors (Basel, Switzerland)*, 16, 2016.

[JBS+10] Jasper Jahn, Ulrich Batzer, Jochen Seitz, Lucila Patino-Studencka, and Javier Gutiérrez Boronat. Comparison and evaluation of acceleration based step length estimators for handheld devices. In *2010 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–6, 2010.

[KGR21] Tim Kuhlmann, Pablo Garaizar, and Ulf-Dietrich Reips. Smartphone sensor accuracy varies from device to device in mobile research: The case of spatial orientation. *Behavior Research Methods*, 53(1):22–33, Feb 2021.

[Lev61] Howard Levene. Robust tests for equality of variances. In *Contributions to probability and statistics. Essays in honor of Harold Hotelling*, pages 279–292, 1961.

[MG22] D. Menaka and Sabitha Gauni. An energy efficient dead reckoning localization for mobile underwater acoustic sensor networks. *Sustainable Computing: Informatics and Systems*, 36:100808, 2022.

[MMM12] C. B. Madsen, J. B. Madsen, and A. Morrison. Aspects of what makes or breaks a museum ar experience. In *2012 IEEE International Symposium on Mixed and Augmented Reality - Arts, Media, and Humanities (ISMAR-AMH)*, pages 91–92, 2012.

[MMT+08] T. Miyashita, P. Meier, T. Tachikawa, S. Orlic, T. Eble, V. Scholz, A. Gapel, O. Gerl, S. Arnaudov, and S. Lieberknecht. An augmented reality museum guide. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 103–106, 2008.

[SBB+15] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Prentow, Mikkel Kjærgaard, Anind Dey, Tobias Sonne, and Mads Jensen. Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. pages 127–140, 11 2015.

[Shc15] Maxim Shchekotov. Indoor localization methods based on wi-fi lateration and signal strength data collection. In *2015 17th Conference of Open Innovations Association (FRUCT)*, pages 186–191, 2015.

[Spe08] *Spearman Rank Correlation Coefficient*, pages 502–505. Springer New York, New York, NY, 2008.

[SSPT18] Rahul P Suresh, Vinay Sridhar, J Pramod, and Viswanath Talasila. Zero velocity potential update (zupt) as a correction technique. In *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–8, 2018.

[SW65] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[TAD+21] Nestor Michael Tiglao, Melchizedek Alipio, Roselia Dela Cruz, Fawaz Bokhari, Sammia Rauf, and Saad Ahmad Khan. Smartphone-based indoor localization techniques: State-of-the-art and classification. *Measurement*, 179:109349, 2021.

[Wei02] Harvey Weinberg. Using the adxl202 in pedometer and personal navigation applications. *Analog Devices AN-602 application note*, 2(2):1–6, 2002.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Game Technology Center

Master's Thesis Project Description

Pathfinding and Localization in AR from Floor Plan Data

Tamara Gini

Introduction

Many mobile apps would like to offer navigation in real-world environments. Creating maps and developing navigation algorithms is challenging especially in complex indoor settings. The goal of this thesis is to use a set of floor plan SVGs as a basis for performing interactive navigation. The implementation part consists of a Web Editor to easily author the extracted floor plan data and a Mobile App to interactively locate and guide users through a real-world environment.

Task Description

Related work research. Explore existing methods and implementations for pathfinding and navigation especially in real-world environments and in AR. **Web Editor.** Create a web editor, which aids the user in authoring the extracted floor plan data such that it can be used for navigation. The server should provide an API, which finds an optimal path between two given locations in the indoor environment. **AR App.** Create a mobile AR app, which uses the pre-authored data to guide a user through the building. The current position should be estimated by using the device's sensors. **Evaluation:** Evaluate the Web Editor and the AR App by conducting a user study.

Milestones

Description	Date
Thesis start	09 Nov 2022
Related work research completed	30 Nov 2022
First prototype ready	01 Feb 2022
Midterm presentation	10 Feb 2022
Implementation complete	24 Mar 2022
Evaluation completed	24 Apr 2023
Hand-in final report	24 May 2023
Thesis presentation	31 May 2023

Remarks

A written report and an oral presentation conclude the thesis. The thesis will be overseen by Prof. Robert W. Sumner and supervised by Börge Scheel and Henry Raymond.