ARCampus Navigation

Praxisprojekt

by Oliver Scheibert

for the

TH Köln – University of applied science

Campus Gummersbach

Faculty of Computer Science and Engineering Science

degree program
Computer Science (BA)

Gummersbach, March 2020

1 Introduction

ARCampus is an ongoing project of the TH Köln – University of Applied Science to explore the potential of augmented reality in several different contexts. This work was done as part of this project.

Augmented reality (AR) has left Gartner's Hype Cycle for Emerging Technologies in 2019[1]. Recent advancements in mobile phone technologies have enabled it to be more accessible to the broad market and be used in productivity. For example, medical AR tools or Military AR applications are already in use with more tools still in development[2] [3].

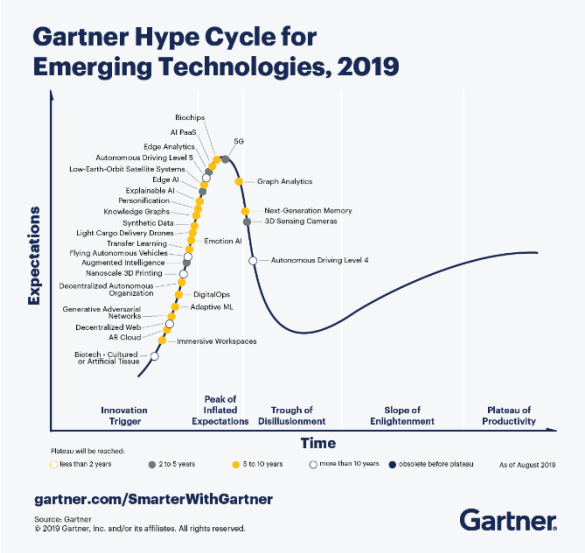

Figure 1: Gartner Hype Cycle for Emerging Technologies, 2019. Augmented reality has left the Trough of Enlightenment and is no longer visible on the chart.

But why augmented reality? It offers a powerful user interface for location-aware computing by overlaying spatially registered, digital information on the user's experience of the world. Usually in real-time to allow the user to interact with the physical and virtual environments instantaneously[4] [5]. This is typically done by either projecting light onto the surface of objects or with see-through devices that can manipulate a video feed or use partially transparent materials to display the digital information[6]. This work uses a smartphone as a video see-through device with Unity and ARCore to calculate the digital information required for this process. Building an AR navigation system with the help of Unity provides a very easy entry into the field but at the same time a challenging set of problems that are deeply incorporated into AR.

The purpose of this work is to test the capabilities of Google's AR tool, ARCore, in the context of pedestrian navigation within a building. An application has been developed

[1] "5 Trends Appear on the Gartner Hype Cycle for Emerging Technologies, 2019," accessed March 6, 2020, //www.gartner.com/smarter-withgartner/5-trends-appear-on-the-gartner-hype-cycle-for-emerging-technologies-2019/.

[2] Mark A. Livingston, Zhuming Ai, and Jonathan W. Decker, "Human Factors for Military Applications of Head-Worn Augmented Reality Displays," in *Advances in Human Factors in Simulation and Modeling*, ed. Daniel N. Cassenti, Advances in Intelligent Systems and Computing (Springer International Publishing, 2019), 56–65.

[3] "FDA Approves AR Surgery Tool That Gives Surgeons 'X-Ray Vision,'" Digital Trends, December 24, 2019, https://www.digitaltrends.com/cool-tech/xvision-spine-system-ar/.

[4] Ronald T. Azuma, "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments* 6, no. 4 (August 1, 1997): 355–85, https://doi.org/10.1162/pres.1997.6.4.355.

[5] J. Grubert et al., "Towards Pervasive Augmented Reality: Context-Awareness in Augmented Reality," *IEEE Transactions on Visualization and Computer Graphics* 23, no. 6 (June 2017): 1706–24, https://doi.org/10.1109/TVCG.2016.2543720.

[6] Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst, "Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR," in *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality* (2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR), Cambridge, UK: IEEE, 2008), 193–202, https://doi.org/10.1109/ISMAR.2008.4637362.

using a model-view-presenter (MVP) software architecture pattern that was implemented within Unity. A virtual model of the main building of campus Gummersbach (Figure 4) has been created which is used to calculate navigation related data, such as paths or destination locations, as well as generate AR guiding elements that are incorporated with the camera feed. To acquire the initial position and rotation (pose) of the user within the building, a marker-based approach is used that relies on detecting the room name from glass plates that are located besides every door within the building.

After the initial localization, users can choose a room within the building and start their navigation towards it. A line will be displayed on the ground that leads to the destination with occasional text prompts to help users navigate safely across floor levels.

2 Related Work

There have been made many advancements in the field of augmented reality and this section will highlight a few that either have an impact on the outlook of this work or that show the current state of the art regarding functionality or usability within that topic.

Yu et al. have developed a mobile application to display information about the flora and fauna found within their campus. One reason behind their choice of using augmented reality is, that through more enjoyment, interest, and a vivid experience with the technology, users are more motivated to visit their campus again. Tracking the position of the user and displaying information using AR technologies are one of the biggest challenges when developing an AR application. Yu et al. have chosen a marker-based approach to track the initial position of the user. By using a combination between: GPS, the electronic compass incorporated within smartphones, and strategically placed markers across the campus, it is possible to track users and provide them with information about the flora and fauna of the campus. This can include a picture of birds and their names, or the name and growth habits of plants and other miscellaneous information related to them. Depending on the amount and type of information they want to display, they can choose between 2D UI-elements or 3D animations and videos to engage users. An example of the interaction between the two methods is a radar with dots on it representing the user's position and points of interest around them and a marker within the video feed that correlates these dots (Figure 2)[7].

Figure 2: UI of an AR campus navigation system by Yu et al. On the top left corner visible is a radar. The marker in the middle corresponds to a dot on the radar.

Another work that focuses on handheld augmented reality introduces a framework and multiplayer engine for AR with treasure hunt games built on top of it. Schmalstieg et al. have developed a marker-based system that introduces augmented reality to museums in the form of puzzles that the users must solve in order to learn about the exhibitions. They have discovered that the average user is not familiar with marker tracking and need time to get acquainted with the interface technology. To elevate this, providing an explanation on the next steps and giving feedback on the scanning process can greatly improve the user's performance with the system. They argued that a natural feature tracking solution which doesn't rely on markers would be better suited for their environment. Their problem is that markers cannot be attached to historical artifacts but still need to be big enough and in view of the camera. Different lightning conditions, such a dim light or reflections

[7] K. Yu et al., "A Mobile Application for an Ecological Campus Navigation System Using Augmented Reality," in *2015 8th International Conference on Ubi-Media Computing (UMEDIA)*, 2015, 17–22, https://doi.org/10.1109/UMEDIA.2015.7297421.

from windows and showcases can hinder the performance of a detection system. Especially exhibitions that can only be shown under restricted lighting conditions make marker tracking unsuitable for their purposes. Since mobile devices did not have enough computational power to support natural tracking at the time of writing their work (2007), they could not implement this technology[8]. 13 years later, ARCore has been developed which proves that the smartphone technology today has reached a point where such a thing is feasible[9].

A novel design for augmented reality interfaces to support indoor navigation, introduced by Mulloni et al, combines activity-based navigation with sparse localization elements to improve user experience and deliver a robust navigation experience. As shown in Figure 3, their interface is divided into two main areas: one that shows activities that the user must perform to reach their goal, the other a video with AR elements to also provide feedback on their next step.

Figure 3: Activity-based navigation with AR supporting elements in the form of arrows.

When walking, users need to divide their attention between the path in front of them and the application. The interface adapts its readability and information density to minimize the information overload they face. This adaptation to the user's attention resources helps them perform better with the system. Their tracking system, which is a marker-based approach, was also improved by using markers in the form of posters that are located on the ground. By providing interesting information, users are more inclined to stand still and interact with the poster. They can be used to generate data about the location of users and signal the system that they might be standing still so that the interface can adapt to the change in attentional resources. They also inform the navigation system of a new position so that it can generate new activities for the user[10].

[8] Dieter Schmalstieg and Daniel Wagner, "Experiences with Handheld Augmented Reality," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, 3–18, https://doi.org/10.1109/ISMAR.2007.4538819.

[9] "Supported Devices | ARCore," Google Developers, accessed April 21, 2019, https://developers.google.com/ar/discover/supported-devices.

[10] Alessandro Mulloni, Hartmut Seichter, and Dieter Schmalstieg, "Handheld Augmented Reality Indoor Navigation with Activity-Based Instructions," in *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11 (New York, NY, USA: ACM, 2011), 211–220, https://doi.org/10.1145/2037373.2037406.

3 Navigation

A key aspect of the application is the navigation system. It provides context in which paths are available to the user that can be queried for the positioning of AR elements within a model (Figure 4) to guide them towards their destination. There are several different methods to display the path data or provide the user information that can help guide them towards their goal.

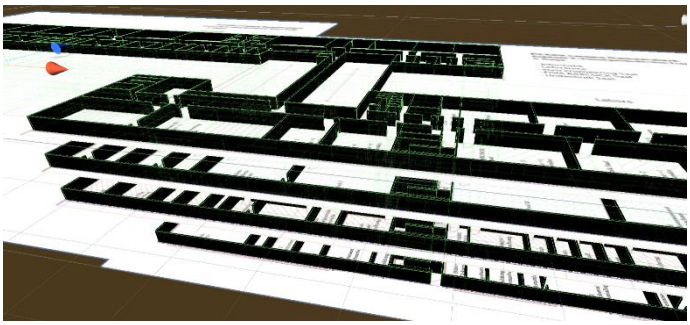

Figure 4: The model of the building with the walls in black and the floors in white.

One method is turn-by-turn instructions[11]. These instructions include visual and audio prompts that tell the user what physical action to take next. “Walk 10 steps forward” or “Turn left” are examples of easy to understand instructions that don’t require contextual awareness of the building around them. The advantage of this is the ability to reduce the impact of low tracking accuracy by providing a unique context with which to think in that doesn’t rely on a map or other model of the building. Using these instructions in an AR context can also mean projecting the UI elements on the floor and walls directly in front of the user without regarding their current pose within the building. A disadvantage of this system is that once a user is lost, due to either too much tracking error or other accumulated user errors, it is impossible for them to reorient themselves. The lack of a reference frame of the whole building or surrounding area that could help them would have to be included separately[12].

Another presentation method of navigation-relevant data is World In Miniature (WIM) models[13]. They are small-scale 3D models of the surrounding area that can be displayed and interacted with using AR tools. By locking the rotation of this model to the rotation of the building and showing the position of the user within, WIMs can provide a better understanding of the larger context of the environment around them. A limitation of the human sensor system is that it can only provide feedback on the imminent surroundings and not on the larger context around. By providing a WIM, the user can compare this model to their surroundings and get information about their current pose and help them

[11] Axel Nix and Andrew W. Gellatly, Turn-by-turn navigation system with enhanced turn icon, United States US7546207B2, filed May 21, 2004, and issued June 9, 2009, https://patents.google.com/patent/US7546207B2/en.

[12] Hee Jeong Kim, Turn-by-turn navigation system and next direction guidance method using the same, United States US7844394B2, filed July 16, 2004, and issued November 30, 2010, https://patents.google.com/patent/US7844394B2/en.

[13] Richard Stoakley, Matthew J. Conway, and Randy Pausch, “Virtual Reality on a WIM: Interactive Worlds in Miniature,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’95 (Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995), 265–272, https://doi.org/10.1145/223904.223938.

identify potential tracking errors. The disadvantage of this system is that it requires a lot of cognitive resources to understand the information displayed and relate it to the surrounds especially when used to counteract these error.

The approach chosen in this work is using NavMesh to generate path data and display it as a continuous line on the ground. NavMesh is a tool found within Unity to calculate a navigation mesh based on a 3D model. This mesh can be queried for information such as the path from a user towards their goal or which terrain the user is currently in. Using a continuous line as the navigation display method allows for an overview of the whole path and provides some feedback of the tracking state by showing the difference in rotation between the line and the orientation of the aisle. As explained in section 5, the current tracking technique is not accurate enough to provide an accurate navigation path. Using lines, this small accuracy can be manually adjusted. This approach is not an ideal approach but provides enough information to evaluate ARCore and analyze the impact of an inaccurate tracking technique on the navigation system.

4 User localization

Determining the position and rotation of a user within a known environment is one of the most crucial steps for the navigation system. Not only is it required to calculate the path from the user towards their destination, it also is used for calculations done by the augmented reality UI system. The positioning system is responsible for placing users within a virtual model of the building and aligning them within the world coordinate space of both ARCore and Unity. The AR elements are then calculated with data from a virtual model and the pose of the user within it. Using the same information base for the navigation system and the AR system promises greater consistency in calculating a path and displaying navigation elements to the user.

The limitation of ARCore is that it only tracks the users pose relative to a starting point. Since this point is unknown, a tracking system capable of localizing the user within the building is necessary. There are several different tracking methods that can be used in an indoor environment. Using beacons, image marker, or text detection methods can be used to calculate the pose of the user. Since ARCore can track the movement of the device relative to that point in space, they only need to be used sporadically.

Beacons: Beacons are known points in an environment that constantly send out a signal. Wi-Fi beacons, such as internet routers, can be located using Wi-Fi antennas which are included in all current smartphones. It is possible to triangulate the position of the user by calculating the distance to three routers based on the time the Wi-Fi signal takes to travel to the users' device. However, walls, the human body, and any solid or liquid matter can interfere with the signal and slow the electric waves down, increasing the time they need between sending and arriving at the user. To combat these variables, error detection techniques can be applied that often require time and sometimes movement of the user within the building[14] [15] [16] [17].

Image recognition: Searching for known visual data within the camera feed or telling users to search for a specific image is another option to acquire their pose[18] [19]. The position of each light ray arriving on the physical camera sensor can be used to calculate the

[14] Joydeep Biswas and Manuela Veloso, "WiFi Localization and Navigation for Autonomous Indoor Mobile Robots," in *2010 IEEE International Conference on Robotics and Automation*, 2010, 4379–84, https://doi.org/10.1109/ROBOT.2010.5509842.

[15] Zhi-An Deng et al., "WiFi Positioning Based on User Orientation Estimation and Smartphone Carrying Position Recognition," Research article, Wireless Communications and Mobile Computing, 2018, https://doi.org/10.1155/2018/5243893.

[16] Jahyoung Koo and Hojung Cha, "Localizing WiFi Access Points Using Signal Strength," *IEEE Communications Letters* 15, no. 2 (February 2011): 187–89, https://doi.org/10.1109/LCOMM.2011.121410.101379.

[17] Yuxiang Sun, Ming Liu, and Max Q.-H Meng, "WiFi Signal Strength-Based Robot Indoor Localization," in *2014 IEEE International Conference on Information and Automation (ICIA)*, 2014, 250–56, https://doi.org/10.1109/ICInfA.2014.6932662.

[18] G.K.H. Pang and H.H.S. Liu, "LED Location Beacon System Based on Processing of Digital Images," *IEEE Transactions on Intelligent Transportation Systems* 2, no. 3 (September 2001): 135–50, https://doi.org/10.1109/6979.954547.

[19] Gianfranco Forlani, Riccardo Roncella, and Carla Nardinocchi, "Where Is Photogrammetry Heading to? State of the Art and Trends," *Rendiconti Lincei* 26, no. 1 (June 1, 2015): 85–96, https://doi.org/10.1007/s12210-015-0381-x.

3D coordinates of the image relative to it. To scan room plates, users must point their devices towards them, which in turn also provides information about the position and rotation of the user during this process. It is important to choose an image that doesn't change frequently itself or its pose over time, such as posters. The room plates that are located on the walls besides every door within the building are the best candidate for this approach.

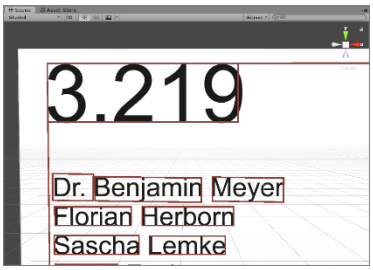

Figure 5: Text recognition result with the Google Cloud Vision OCR service. The lines are boundaries of each detected word within the picture.

ARCore delivers image recognition technology in the form of "Augmented Images". By providing the software with a set of images of room plates, it is possible to query the current video feed for a feature match and track the pose of the image relative to the smartphone. The absolute pose of the user can be calculated using this information and information from a marker database that contains the pose of the marker within the building model described in section 1. A problem with using Augmented Images is that the room plates in the building are made of glass which leads to inconsistent image quality under different lightning conditions. Reflections and shadows greatly alter the image to the point where the reference image cannot be found and ARCore is unable to calculate a pose for the room plate. Even under ideal lightning conditions, many of the room plates don't have a lot of text that would generate feature points. If an image has too few of them, ARCore might even be unable to track the image at all.

Text detection: To evaluate the position of the room plates without needing to generate new images every time they change, only the text can be evaluated. It is very common for rooms to have different people inside, which require an updated plate. Since the name (number) of each room stays consistent, only that is required to match a plate with a position in the marker database. The room information is stored inside a JSON, which makes it easy for future updates. Generally, optical character recognition (OCR, Figure 5) systems also perform better than ones that rely on feature point detection. Since no reference image is needed, texts can be analyzed under different lighting conditions. As long as the text is readable by a user, an OCR system should be able to also deliver a result. A disadvantage of this approach is that the pose of the user can only be approximated. Especially the rotational values during the scanning process are prone to error. During the software tests, it became clear that the rotation was often offset by over 10°. This required a manual adjustment tool to change the rotation by hand. Due to a lack of time resources, another approach could not be evaluated. Even then, the current state of the system is enough to evaluate the continuous tracking capabilities of ARCore within an indoor navigation context.

5 Continuous Tracking

Tracking is an important part in any AR application. It dictates how well the virtual information can be overlaid on top of real objects. Over the past decades, there has been made a lot of research into tracking methods, such as inertial- to optical-sensing techniques[20]. Each technique that relies on different sensors brings their own advantage regarding computational requirements and accuracy. Modern cellphones have a host of sensors, like accelerometers, gyroscopes and compasses which internally perform sensor fusion. This allows them to report their absolute orientation in the world coordinate frame as a 3 x 3 rotation matrix, containing Euler angles for yaw, pitch and roll[21]. In addition to that the Android operating system also reports the accuracy status of this result. This rating ranges from "unreliable" to "accuracy high"[22]. ARCore uses this data to combine it with feature points extracted from processing the camera feed to constantly calculate the pose of the user[23].

ARCore tracks the user pose relative to a point in space where the tracking process started. Using the text detection as mentioned in section 4, this point can be calculated at the start or during the navigation. The pose data from ARCore is used to move a virtual user within the virtual building model to calculate the navigation information, such as AR elements. It became apparent during the testing that ARCore was not built for an indoor navigation system. The tracking accuracy drops dramatically over time and distance traveled. Even basic movements such as a 180° turn are problematic during longer navigation periods. To combat the impact on the user experience, they have to be informed of the current error state of the system. The approach chosen here was a change in AR elements color over time. Future works might go into more detailed error tracking and visualization techniques.

While briefly mentioned, another issue is the small amount of information about the error rate of either the smartphone sensors or ARCore. It is almost impossible to inform the user of the system state without an accurate understanding of its current error accumulation state[24]. MacIntyre et al. believe that the tracking problem, which is the accumulation of registration error over time, will not be solved in the near future. Which is why the AR

[20] G. Welch and E. Foxlin, "Motion Tracking: No Silver Bullet, but a Respectable Arsenal," *IEEE Computer Graphics and Applications* 22, no. 6 (November 2002): 24–38, https://doi.org/10.1109/MCG.2002.1046626.

[21] V. Bettadapura, I. Essa, and C. Pantofaru, "Egocentric Field-of-View Localization Using First-Person Point-of-View Devices," in *2015 IEEE Winter Conference on Applications of Computer Vision*, 2015, 626–33, https://doi.org/10.1109/WACV.2015.89.

[22] "Sensors Overview | Android-Entwickler," Android Developers, accessed February 25, 2020, https://developer.android.com/guide/topics/sensors/sensors_overview?hl=de.

[23] "Fundamental Concepts | ARCore," Google Developers, accessed February 29, 2020, https://developers.google.com/ar/discover/concepts?hl=de.

[24] B. MacIntyre, E.M. Coelho, and S.J. Julier, "Estimating and Adapting to Registration Errors in Augmented Reality Systems," in *Proceedings IEEE Virtual Reality 2002*, 2002, 73–80, https://doi.org/10.1109/VR.2002.996507.

system should provide feedback in regards to its accuracy, so that the application can react to it. ARCore already does calculations on this error rate, but only provides scarce feedback on them in the form of a tracking state. Which only dictates if the tracking result is "reliable" or not. A more precise feedback or error prevention system could be integrated with UI elements, as explained in section 7.2[25] [26].

For the purpose of this work, the error rate is assumed to be in direct relation to time and distance to the last known (marker) location[27]. The predicted angular deviation over time of gyroscope sensors found in smartphones is predicted to be less than 0.1 degrees within 10 seconds of use[28]. A threshold of 100 seconds has been chosen after which to warn the user of an unusable tracking state. The tests in section 10 have shown that after 120 seconds of use, ARCore can already show tracking drift. A change in rotation of more than one degree also affects the user experience negatively when navigating through the building. When the tracking state is deemed unusable, the user has to scan a room plate again and reset the accumulated errors as much as possible.

5.1 ARCore tracking

ARCore provides an inside-out tracking system through concurrent odometry and mapping (COM)[29]. Using inertial measurements units (IMUs), such as gyroscopes, magnetometers, and accelerometers to calculate the motion of the smartphone and combining that with feature points which were extracted from the camera feed, allows for a tracking method that can correct the registration error of these sensors. A virtual map is generated that combines all the knowledge provided by the COM and places the phone within that map. For each set of feature points that create a representation of the surrounding world, a new map is created and if several of these maps overlap, they combine into a bigger one[30]. This approach works best in smaller environments and can introduce problems when used in larger ones, especially when they don't provide a significant amount of unique feature points. During the development of this application, there have been several occasions where the tracking suddenly jumped locations. This is most likely due to the

[25] DREXEL HALLAWAY, STEVEN FEINER, and TOBIAS HÖLLERER, "Bridging the Gaps: Hybrid Tracking for Adaptive Mobile Augmented Reality," *Applied Artificial Intelligence* 18, no. 6 (July 1, 2004): 477–500, https://doi.org/10.1080/08839510490462768.

[26] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu, "Analysis and Modeling of Inertial Sensors Using Allan Variance," *IEEE Transactions on Instrumentation and Measurement* 57, no. 1 (January 2008): 140–49, https://doi.org/10.1109/TIM.2007.908635.

[27] MacIntyre, Coelho, and Julier, "Estimating and Adapting to Registration Errors in Augmented Reality Systems."

[28] Anton Umek, Anton Kos, and Sao Tomaic, "Validation of Smartphone Gyroscopes for Angular Tracking in Biofeedback Applications," in *2015 International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI)*, 2015, 294–98, https://doi.org/10.1109/IIKI.2015.70.

[29] "Fundamental Concepts | ARCore."

[30] Esha Nerurkar, Simon Lynen, and Sheng Zhao, System and method for concurrent odometry and mapping, United States US20170336511A1, filed May 15, 2017, and issued November 23, 2017, https://patents.google.com/patent/US20170336511A1/en.

repetitiveness of aisles within the building. When calculating feature points for these aisles, it is possible that there can be a match between two different sections of the building, resulting in a false pose correction.

5.2 Error accumulation in smartphone sensors

Even if the user's pose is known at one point, when sensor accumulate too many errors over time, the pose estimation that relies on them also becomes unreliable[31]. Not only does the data from gyroscopes deviate over time, as described above, so do other sensors. Magnetometers exhibit non-uniform errors over their tracking space if presented by a strong magnetic field. These can come from magnets or even human bodies[32]. Another issue is that the time between sampling a sensor and calculating the input can vary due to different system latencies or CPU loads. This calculation offset also introduces geometric errors over time[33].

Since these errors are unavoidable, they have to be accommodated for by the system through either UI implementations or predictive algorithms that can counteract them. Since the most common error can be described as a function over time, the current solution to keep track of these error is also using time[34]. In the future, a more advanced error tracking system is required. Using a combination of different continuous tracking technique instead of just relying on ARCore's to create a hybrid tracking system would allow for a more robust system[35]. A system that can also be queried for non-linear in addition to linear errors and accommodate for them.

[31] El-Sheimy, Hou, and Niu, "Analysis and Modeling of Inertial Sensors Using Allan Variance."

[32] B. MacIntyre and E. Machado Coelho, "Adapting to Dynamic Registration Errors Using Level of Error (LOE) Filtering," in *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, 2000, 85–88, https://doi.org/10.1109/ISAR.2000.880927.

[33] MacIntyre, Coelho, and Julier, "Estimating and Adapting to Registration Errors in Augmented Reality Systems."

[34] MacIntyre, Coelho, and Julier.

[35] HALLAWAY, FEINER, and HÖLLERER, "Bridging the Gaps."

6 Implementing a model-view-presenter architecture within Unity

One goal of this work is to provide a general software structure for an augmented reality navigation system. It will be used to test ARCore as a tracking and NavMesh as a navigation tool to explore UI implementations for AR applications. The following sections will answer the question on how to implement the MVP pattern within Unity and its compatibility with ARCore and NavMesh.

6.1 Using Unity

Unity is one of the most popular software for augmented/virtual reality content development in 2019[36]. The game engine supports several different AR tools, making it a valid option for approaching the technology from different directions and with different motivations[37]. Surrounding Unity is a vast eco system of information in the form of tutorials, forum posts and developer guides that acted as an important resource during the development process of this software. In its core, it is a component-based system that also provides an editor which acts as an interface to show the relation between GameObjects. These are the foundation of the engine and act as container for different components or scripts. For example, to form a cube, components with information about its position, rotation and size are added to a GameObject. If the cube should move by its own, a script can be added that regularly changes the position value of the corresponding component. Unity also provides a physics engine that can translate a speed value to a change in position over time. It tracks physical boundaries of objects so that walls or floors can be created that other objects can't pass through. 2D UI elements are usually rendered on a canvas, which is a representation of the phone display. The position and size of these elements are also determined by components. To display AR elements, ARCore provides a virtual camera that moves within the Unity world coordinate system based on the smartphone movements. Its virtual properties are similar to the phone's camera so that GameObjects that are placed in front of it can be directly rendered onto the camera feed.

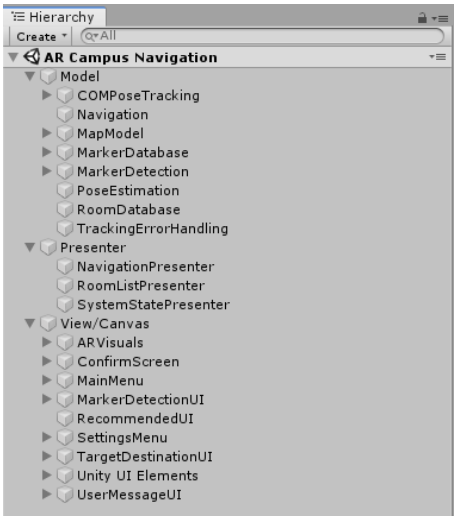

Figure 6: Unity component hierarchy modeled after the MVP pattern.

[36] "Unity Public Relations Fact Page," Unity, accessed April 21, 2019, https://unity3d.com/public-relations.

[37] "Get Real with Creating AR Games and Apps in Unity," Unity, accessed March 5, 2020, https://unity3d.com/how-to/create-AR-games-in-Unity-efficiently.

Every GameObject can be either constructed dynamically with code or manually within the editor. There, they are organized in “scenes” where their hierarchy is defined as a tree structure. This allows for an instant overview over them that can be expanded to reveal different depths of the structure.

6.2 Functional and non-functional specifications

The following requirements are based on a minimum degree of functionality that this AR navigation software must deliver to provide a functioning system that can be evaluated. It includes specifications for the navigation and tracking system as well as usability of the UI.

Complete model of the main building: The campus Gummersbach has a total of 5 buildings where lectures take place with more coming in the future. This work will only model the main building with its 4 floors. It is the biggest building with most events taking place there, making it the best candidate to test all the capabilities of the application.

Easy expandability of the model: The addition of more buildings to the model should be easy and independent of existing models. Each model should allow for an individual scale and also be independent of each other.

Scanning room plates: Information about the room plates should be automatically acquired without further user input. This information can include: text, position, rotation, or other visually detectable information.

Pose tracking for navigation: To allow for the navigation through the building, the current pose of the user or their smartphone within the building needs to be determined. The tracking error should be kept small enough for the user experience to not suffer from it.

Project navigation information on the ground: Using the pose of the user and the model of the building, realistic looking 3D information should be projected onto the camera feed of the smartphone showing the user different kinds of information. This can include: The path to the destination, information about rooms in the vicinity, information about the building.

Display the navigation accuracy: Smartphone sensors and calculations of the relative pose of AR elements may accumulate errors over time. These errors can reduce the accuracy of the navigation information. Since this is important information for the user to make educated decisions on their action, they should be displayed to them in any way possible.

The app should be safe to use: Navigating through a building brings inherent risks, such as running into other people or falling over obstacles. The app should take the safety of the user into consideration and reduce the risk to the user as best as possible.

Navigating through the building should be easier with the app: The app should not make it harder or more time consuming to locate a destination than current methods.

Rooms are numbered in a logical order to help students find their destination quickly. The first number of the room represents the floor number, the second indicates if the room is on the east or west side of the building. This numerical order is not obvious at first glance and might make navigation harder for visitors that do not know about these rules. To elevate this issue during events where a lot of visitors are expected, signs are placed throughout the whole building as a guiding help. The application should help this process and not make it harder.

6.3 Development environment

The application has been developed and tested with the following tools:

- GitHub
- Unity version 2018.3.14f1
- Huawei P20 Pro, EMUI version 9.1.0 (Android version 9)
- ARCore SDK for Android version 1.15
- ARCore SDK for Unity version 1.15

Almost all of the software versions are the most recent releases except of Unity. Updating Unity can often make parts of the software obsolete or deprecated which might introduce unexpected errors that have been accounted for in the initial version. To keep the development process simple, the most recent Unity version at the beginning of the development work was chosen.

6.4 Architectural approach

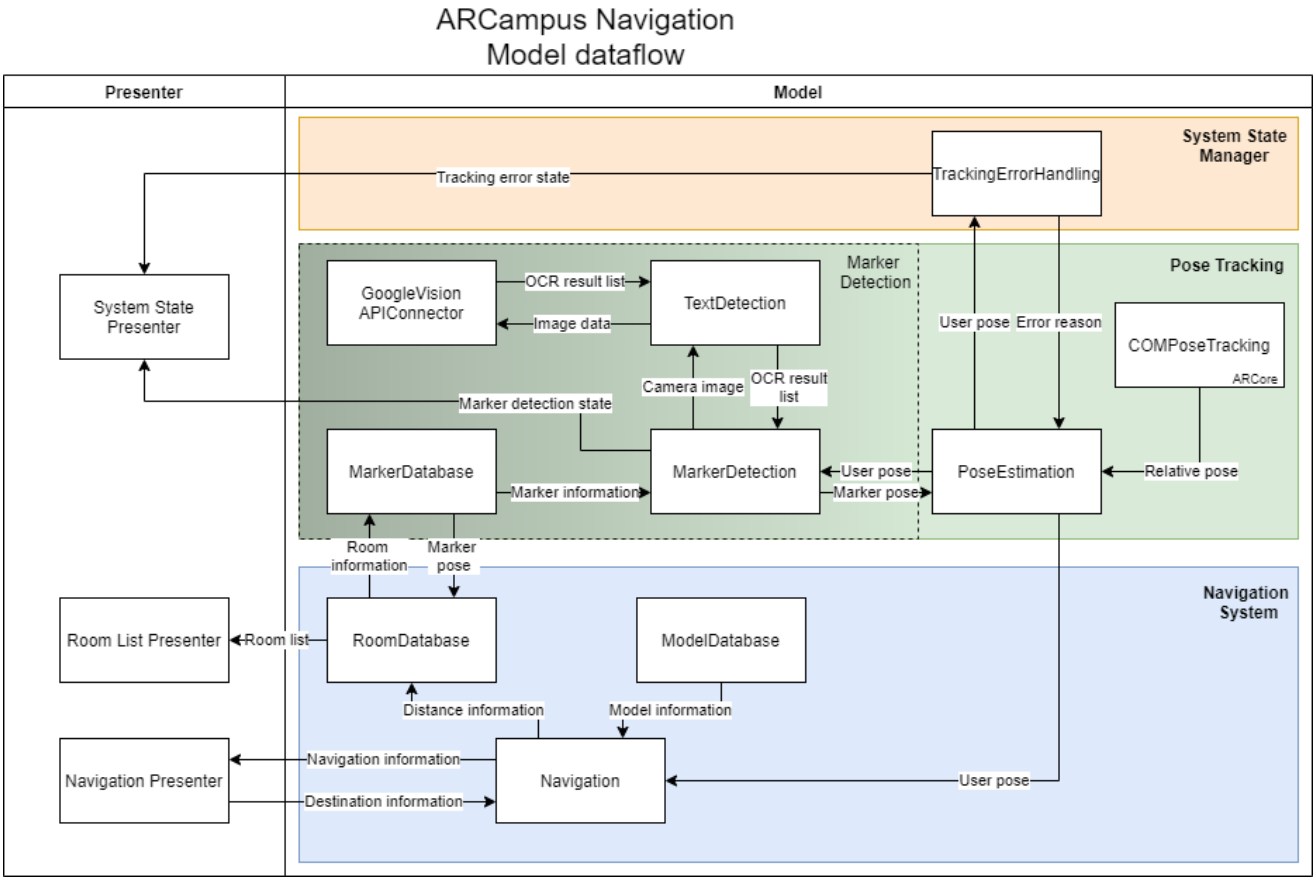

Figure 7: Presenter-Model components. The model is divided into the three categories: System State Manager, Pose Tracking System, and Navigation System. The marker detection system is part of pose tracking and handles text detection and marker pose tracking functionalities. The View components are not visible but follow a similar approach in structure and data flow.

The structure of the architecture is based on a Model-View-Presenter pattern. It divides the software into three parts; the UI (view), the business logic (model) and presenter components to allow communication between view and model. Two of the decision forces before choosing this pattern were the ease of implementation within Unity and the experience of the developer with the pattern.

To take advantage of the Unity UI, the component hierarchy structure is closely modeled after the MVP pattern. As you can see in Figure 7, the components are structured within the previously mentioned hierarchy. This allows for an overview of each component within the software and display their connection between them. When highlighting a component within the editor, the scripts for each class show entries for other classes that get data from the highlighted one.

To take advantage of the rich UI interface implementations that Unity provides, they have to be added separately to the hierarchy. The view GameObject not only acts as an overview of the view software components but also as a container of Unity UI elements in the form of an Unity canvas. This allows for manual adjustment of the menus and navigation display elements by showing the finished UI within the editor (Figure 8).

The advantage of using the Unity editor to display the software structure and use its canvas functionality is that a developer can get a clear overview of the capabilities of the application without having to read any code.

The view elements of the MVP model are not depicted here because this work focuses more on the business-logic aspects of the software. Generally, the view components are divided into 2 categories; one that handles 2D UI elements and another one that is responsible for AR visuals.

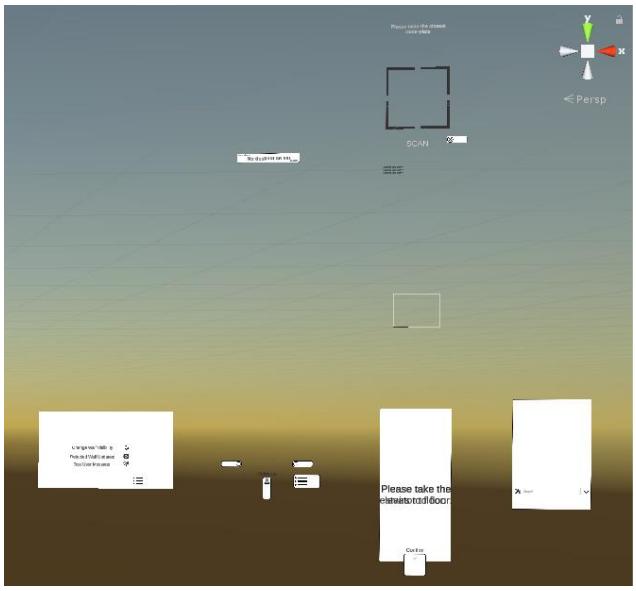

Figure 8: Unity canvas with all 2D UI elements side-by-side.

The following presenter-model components, shown within Figure 7, are the divided into three categories;

System State Manager: This category is responsible for keeping track of the state of the system, such as the AR tracking error (as explained in section 5.2) and the current state of features such as OCR services. Currently, the communication between this manager and the UI is mainly to show the marker tracking state or to display tracking error values.

Pose Tracking System: The pose tracking system contains a central component that takes input from others and combines them to calculate the current user pose within the building. The localization component consists of a marker-based tracking system that relies on text detection to match a marker in front of the camera with one stored inside the marker database. It functions by sending an image from the camera feed to the Google Cloud Vision service and waiting for the response. It usually takes less than one second for the text detection results to arrive, which can then be search for strings containing the room numbers. It is the only part of the software that is connected to the internet.

The other system is the continuous tracking system provided by ARCore. It tracks the user pose relative to the last detected marker. The user pose is stored by moving a virtual user agent within the model and which can be accessed for its position and rotation values when needed.

Navigation System: The navigation system mostly relies on NavMesh. A system that provides navigation data when given a 3D model, a user position, and a location for the destination. Stairs and elevators are marked as special areas within this mesh, so that instructions can be displayed to the user when they enter such an area. The path consists of an array of 3D coordinates on the model that can be used for the navigation UI.

6.5 Navigation with Unity NavMesh

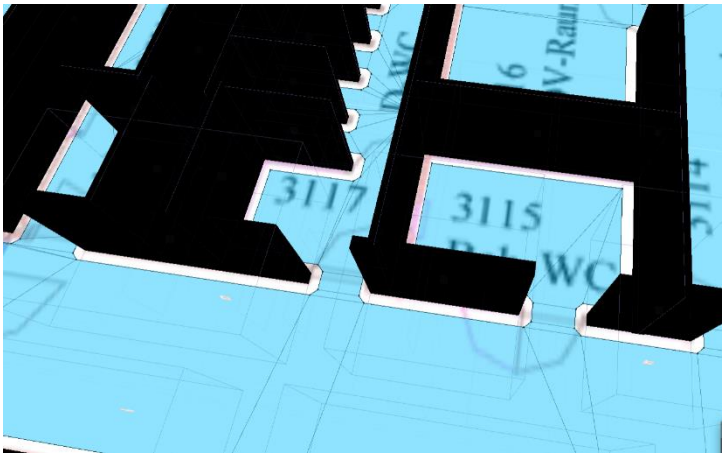

Figure 9: Navigation mesh in blue around the walls of the model. The blue area can be queried to find paths from one location to another one within.

An advantage of using NavMesh is, that it provides a big feature set originally designed for game development. This feature set includes a tool for querying a model for walkability and pathfinding purposes. Unity can calculate a mesh based on a 3D model that can be used for navigation calculations, called NavMesh (Figure 9). This mesh is generated before the compilation of the program which reduces computational requirements of the AR device and allows for error detection early in the development process. For example, the mesh is calculated based on physical data about a virtual user that walks on it. The wider the agent, the harder it is for them to walk into small places.

The Unity NavMesh class automatically calculates and updates the path from the user in the model towards their destination. It provides information about these calculations, such as the total distance and the 3D coordinates of each corner of this path. It is also possible to query the current position of the user for additional information. Stairs, elevators or otherwise dangerous areas can be marked. When users enter these regions, special code can be executed based on which type it is. This is helpful to avoid dangerous situations, such as walking up or down stairs while the user attention is fixated on the smartphone in their hands instead of the stairs. The application will then interrupt the navigation process and ask the user to autonomously traverse the stairs. Once they reach the desired floor, the navigation can continue.

The accuracy of the navigation information is based on the accurate size and proportions of the virtual model and the user pose within that. The initial intention was for an implementation of occlusion effects on the AR elements. But testing this feature has shown that a sudden interruption in the line was very confusing. It outweighed the advantages of an improved immersion. It was also problematic when the pose tracking system failed and often resulted in a complete loss of the navigation data, because it could be hidden behind walls.

7 UI design decisions

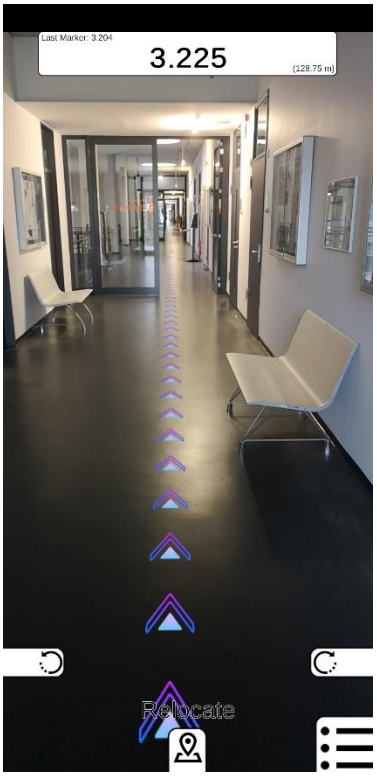

Figure 10: Navigation UI with the destination visible at the top. The arrows on the side are for manually adjusting the user pose. The list icon will show a list of all the rooms (Figure 11).

A big challenge in designing UI elements for video see-through augmented reality devices is to find a balance between immersing the user in the video feed and bringing their attention back to their surroundings. Navigating through a building brings inherent risks with it. Obstacles, like other people, benches, or doors, can surprise an unexpecting user. Not to mention potentially dangerous areas such as around railings in the 3rd floor with a 20 meter drop down behind them. It is important to show the user information that they are searching for while also keeping their attention on their surroundings when they are moving. This balance was one of the driving factors of the UI design of this application. However, it was mainly developed based on intuition and carful consideration because the main focus was on functionality and not an user-centered approach. An in-depth analysis and other forms of potential UX designs for augmented reality is a potential goal of future works using this documentation. A few of the other considerations are listed here:

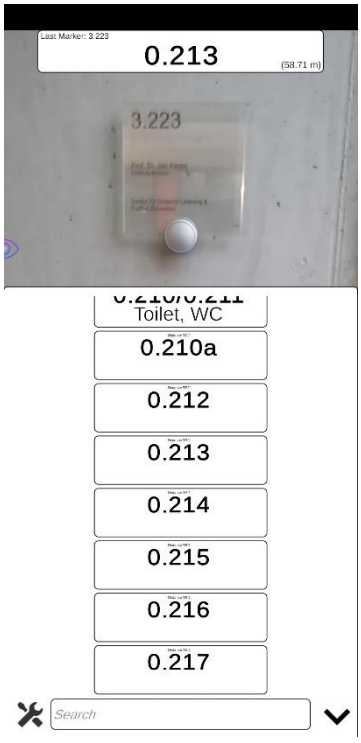

Figure 11: Roomlist UI.

The basic idea behind the duality between displaying 2D UI elements and 3D augmented elements is that if one should be in focus of the user, the other should be more in the background to not draw unnecessary attention. For example, if the user has chosen a destination and is guided towards it by the arrows, the 2D elements for accessing different menus and functions are all located at the edge of the screen with little color or animations (Figure 10). The goal is to make the user focus on their way ahead, either through the camera feed or by looking up. To also make them look up from their phone more and not focus their attention on the device, the 3D elements are relatively simple but easy to distinguish from the ground. The colorful, moving arrows on the ground make it easy to see what the next step is even when just glancing at the screen. When choosing the destination (Figure 11), most of the screen is filled with information and requires a lot of attention from the user. To make them stop in their tracks, using text can also be an option. Since it is hard to read while walking, most user should stand still for the moment and concentrate on the 2D UI.

In this regard, error messages or important alerts also cover the whole screen and need the users' attention before they can resume with using the app. As mentioned before, navigating through a building can bring dangers and to make the user aware of them, it is sometimes important to disrupt the user experience in favor of safety.

7.1 Augmented Reality elements

Since the focus of this work was the implementation of AR elements in a navigation system, the current solution was one that is easily implemented and clearly displays tracking errors. Using the marker-based approach that relies on text scanning, followed by continuous pose tracking with ARCore, means that there are two possible error sources: the immediate error from calculating the rotation of the user when scanning a room plate and the concurrent error caused by ARCore. To display both of them as simple as possible, a line which can change its color has been chosen. Using a line makes it easy to discern the initial rotation error by comparing the displayed path with an aisle. Since the line is animated on top of the floor and shows the path towards the destination, a misalignment can be easily detected. By providing the user with a manual rotation tool, the line can be rotated to better match the direction of the aisle.

The line itself is an array of arrows that are constantly moving towards the destination. The width of the line appears to be bigger starting at the location of the user and smaller in the distance. Which gives it the illusion of existing in the 3D plane of the building. Each arrow on the line is also brightly colored in blue and purple tones. This is to provide more visibility compared to the ground since these colors are less likely to be found in buildings.

7.2 Error visualization

AR navigation visualization elements rely heavily on the tracking accuracy. If the position of the elements doesn't align with the world as shown in the camera feed, navigating through buildings might get too difficult. This effect could be minimized by an appropriate error prevention method as discussed in section 5.2. Another option is for UI elements to show users the rate of reduced accuracy or change the visualization method completely to give them as much information as possible to help them make a decision on what to do next[38].

There are different visualization methods to display the degree of error and accommodate for a reduced tracking accuracy in AR navigation applications. One is to show guiding

[38] Frieder Pankratz et al., "User Awareness of Tracking Uncertainties in AR Navigation Scenarios," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013, 285–86, https://doi.org/10.1109/ISMAR.2013.6671807.

information in the form of an avatar which leads the user towards their goal. It allows for a playful way to hide the exact location of the AR visuals and still help guide the user in a general direction[39]. Another is by using a WIM. It shows the whole building, its rotation and how the user is positioning within that. Users can detect tracking errors by comparing the information from the model with their current situation within the actual building. The disadvantage is that it requires a lot of cognitive resources to understand such a model[40]. In general, it is best to use a visualization technique that shows many different possibilities of where to go next within a short range of possible targets and let the user make the decision of how to proceed. The change in the visualization has to be severe enough that the user understands the degree of tracking error but still allow them to navigate through the building.

Due to time constraints in this project, it was not possible to implement these systems. Instead, an option to display the virtual walls around the user within the building was added to a settings menu, which allows for a comparison of the error state similar to a WIM. The navigation line also provides feedback on error in rotation, and when reaching the destination, on the position of the user.

[39] Pankratz et al.

[40] Stoakley, Conway, and Pausch, "Virtual Reality on a WIM."

8 Difficulties

During the development of the application, there have been many different approaches to solving the tracking issue. Explicitly mentioning them in this section is meant to provide information for future works that might have the same issues.

8.1 Initial positioning

The first part of the research was to find a robust localization method to provide the user's pose to the AR system. Since most of these systems rely on either an optical marker-based approach or a beacon focused strategy, both have been evaluated. Using beacons usually includes either dedicated systems that were set up specifically for the application or rely on using existing beacons, such as Wi-Fi routers. Installing new systems, like infrared beacons, would have not been feasible considering the size of the building and their cost, initially and in maintenance. Wi-Fi routers were a great contender for such a beacon system because they are already installed within the building and are constantly, and publicly, emitting a signal that can be read by most modern smartphones. However, with the introduction of Android 9, restrictions regarding permissions and frequency of Wi-Fi scans were introduced; A foreground application is only allowed to scan four times in a 2-minute period[41]. This severely restricts Wi-Fi scans capabilities in using it for continuous tracking systems that rely on frequent scans. Also, with an estimated mean error of 2.3 m to 2.9 m, the accuracy of using Wi-Fi signals is too low to be used as a reliable source of pose data[42].

8.2 ARCore Augmented Images

ARCore provides a feature called Augmented Images. It allows for the scanning of images within the video frame. It is possible to use this feature to scan room plates and track them with very high accuracy. ARCore provides the pose of the image within the video feed relative to the user. This could be used to calculate their exact position and rotation. One of the limitations of this feature is the work required to generate these image targets. The information of the room plates is stored in an outdated format that cannot be easily extracted into an usable image format. Additionally, the unique feature points of each image are not enough for ARCore to reliably track these images. When added to the image database, they get rated by ARCore on how well they perform as an image target. The room plates of floor 3 usually are rated at 0..20 points out of 100. The last disadvantage

[41] "Wi-Fi Scanning Overview | Android-Entwickler," Android Developers, accessed March 2, 2020, https://developer.android.com/guide/topics/connectivity/wifi-scan?hl=de.

[42] Atreyi Bose and Chuan Heng Foh, "A Practical Path Loss Model for Indoor WiFi Positioning Enhancement," in *2007 6th International Conference on Information, Communications Signal Processing*, 2007, 1–5, https://doi.org/10.1109/ICICS.2007.4449717.

of this feature is that the room plates within the building are made of glass panels. Tracking them during different lighting conditions has proved to be sometimes impossible. The reflections shown on the glass are much more visible on the camera than on the human eye. Because of the unreliability of this approach, it was deemed unusable.

8.3 Line detection to improve tracking accuracy

An issue with the current marker-based tracking method is that it does not provide accurate rotation data. A potential fix for that would be to scan the video image for lines that could be mapped to the floors within the building. Using line detection methods, such as a Hough line transform combined with the edge detection system used by ARCore might prove promising results[43]. It was clear that the high resolution of the test device's camera, 7360 x 4912 pixels, meant that calculating lines for each pixel would consume too much CPU resources. To evaluate other line detection methods or to optimize this approach exceeded the purpose of this work and would have taken too much time and resources. It is a topic for future works that focus on a reliable tracking system for indoor navigation.

[43] Chengping Xu and S.A. Velastin, "A Hough Transform with Integral Kalman Filter Refinement," in *IEE Colloquium on Hough Transforms*, 1993, 4/1-4/4.

9 Maintenance

Maintenance will be an important step for the application to improve further and make it available for future use. Others should be able to profit from the information base acquired with this work and produce improvements without too much overhead. The biggest disadvantage of handing off a software project is the lost information that has not been documented or is inherit in the intuition of the original creator. The purpose of this section is to fill in any gaps of information needed between what is described above and what is required for other developers to continue working on this project.

Adding new models is important for the future of the project because it should be easy to adapt this navigation tool to any building and any number thereof. Changes in the model structure itself should not be necessary but rather the approach to add them. The current method is to add them wall-by-wall based on the floorplan of each floor. This is tedious work and could profit from an automated process. Models created by almost any tools that generate a 3D model, can be imported into Unity and used for the navigation system.

9.1 Building a Unity model

The model used to determine the path from the user to their destination is based on a floor plan of the building. The plan was added as a texture to a 3D plane and scaled as needed. Since in Unity's coordinate system one unit equals one meter, the floor plan needed to be manually adjusted and the results tested using ARCore. The walls are GameObjects that are extruded from the walls as shown on the plan. Doors and other more complicated structures have been simplified as they are not needed for the purpose of this work. Stairs and elevators are modeled as virtual connections between two floors using ARCore's mesh linking feature. The area around them is marked to scan for users that enter them and adjust the navigation information in response. After the model is complete, a navigation mesh can be generated. The mesh, as visible in Figure 12, can be manually checked for any missing doors or marked areas.

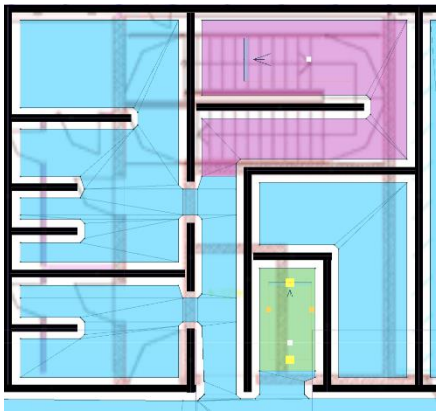

Figure 12: NavMesh with marked areas for the stairs (red) and elevator (green). The little arrows and yellow squares visible within these areas are links that connect to the floor below.

9.2 The model structure within Unity

The model of the building consists of 4 floors that each have a 3D-ground-plane with a texture that contains a picture of the floorplan. The walls are cubes with the width, depth, and height adjusted based on the plan. They need a box collider component attached to

them for NavMesh to calculate the walkable area on the floor plane. They also have to be assigned a “Walls” UI layer so that the ARCore camera can be set to not render them within the video feed. Because the floor plan is divided into east and west, two sets of planes and walls have been created separately, with a third set of GameObjects containing the stairs and elevators. These are modeled as smaller planes within the floor and have a component attached to them that changes the NavMesh area type to a custom “Stairs” or “Elevator” type. To connect two floors together, NavMeshLinks have been used. They are virtual connections that are used for the path calculations. To allow a path to go through the floor, the ground plane should not have a box collider attached. The navigation mesh will still be calculated on top of it. The stairs could also be modeled as physical steps, but since they are not used as a basis for the navigation information, their representation was kept as simple as necessary.

9.3 Updating the room information

A big challenge is to keep the information about the rooms contained in the database up to date. Because of restructuring, the people that work in the rooms and their associated institutes can change several times per year. Changing this information within the application should be easy and not require a change within the data structure itself to keep the maintenance as low as possible. The current method of storing this data is with a JSON file containing the two datasets: “RoomName” and “Description”. The “RoomName” field contains the number of the room. It is important to keep the naming of this field consisted within the app. The name should be a number, or sometimes two, that contains a period instead of a comma. The “Description” field contains any other information about the room, including; the institute name, names of people working in the room, or other descriptive information such as “Toilet”, or “Exit”. This simple structure was made with the intention of keeping it futureproof, without the need to change any part of the code that interacts with it. The stricter rules of the “RoomName” needed to be made in order to identify the locations of the rooms within the model.

9.4 Updating the marker position

The room plates within the building represent the markers for the marker-based approach used for localizing the user. The pose calculation requires the location of the markers within the model. They are modelled as cubes with their GameObject names as the room name and a transform component attached that represents the pose within the model. The general position of the plates is usually besides the room doors, but the exact location had to be manually confirmed. Positioning the marker on the wrong side of the door could lead to an offset of over one meter, which is very noticeable when navigating through the building towards stairs or corners. The current error tolerance of their position is

approximately 20cm. The more accurate the position of these markers within the model, the better the pose tracking accuracy.

9.5 OCR key update

The current OCR method relies on the Google Cloud Vision API, which requires a private API key. A key provided by the university is needed in the future, because the current one has a limited amount of usage per month and limited time of how long it can be used for since the start of development. The key is located in a txt file within the “Resources” folder of the project and accessed during the start of the application.

9.6 UI improvements

The current UI design was made based on the intuition and experience of the developer. Since there has not been a formal approach to this design, it could also not be evaluated. The interaction of the user with the AR elements and their exact design need to be formerly tested and improved on in the future. The current goal was to provide a usable interface that displays necessary information about the destination and tracking accuracy.

10 Software tests

To test the capabilities of ARCore and the navigation system, simple tests were devised. One purpose of these tests is to determine the rate of error ARCore produces and how that might affect the user experience. The other purpose is to determine the effectiveness of the navigation system in guiding the user towards their destination and provide feedback about the path and the state of the system along the way.
The tests are not meant to deliver precise data about these results, but a base understanding on which to progress further when researching in AR user experience or tracking error prevention methods.

10.1 Testing ARCore error accumulation

Early tests into ARCore have shown promising results regarding its ability to keep track of fast movements and movements over a longer period of time. To evaluate these first impressions, here are the test scenarios using ARCore within a completed application that has many calculations running in the background and their potential effect on the error accumulation through limited CPU resources. They were devised to test the limits of ARCore and represent edge cases during an average use case.

Scenario 1: Moving from room 3.225 to room 3.204

Figure 13: Initial rotation error in red and manually corrected rotation in blue.

To test the scale of the model using ARCore's tracking system, the navigation accuracy between only two rooms has been evaluated. If there is an error between the indicator of the destination and the actual room plate either the scale of the model, the tracking accuracy, or both introduced errors. The navigation was started at room 3.225, which is on one side of the building and the destination set to room 3.204, which is on the other but still on the same side.

The test has shown that traveling between the two points introduced an error of 5.5m sideways and 2.75m in length with a navigation time of 2:30 minutes. During the room plate scanning process, a rotational offset has occurred that was manually adjusted, as seen in Figure 13. The adjustment was made so that the guiding line points towards the middle of the aisle when looking from the first room to the second. It is unclear if the consequent sideways error was due to this adjustment or because of tracking issues from ARCore.

Scenario 2: Moving from room 3.101 to room 3.109 and back to room 3.101

This test is to determine the positional drift of ARCore. Navigating from one room to another and back, along a straight corridor to minimize the effect of rotational errors allows to test the error accumulation of the positional data. The accuracy of the model should not play any role in the result because the start and end position are the same.

The distance between the two locations is approximately 130m. Navigating from the initial room to the second posed similar difficulties in adjusting the rotation of the model. During tests when it was possible to reach the turning point, turning around posed another issue. It seems that after a walking duration of 2:30 minutes, or distance of 130 meters, the tracking system had trouble keeping up with a 180° turn. It would always lose the tracking position and randomly pick it up after a few seconds. The old position was never automatically reacquired, and the test could not be continued due to this error.

Scenario 3: Rotating the device for 2 minutes with natural rotation and position changes

After having tested the position error accumulation, the next test is to determine the rotation error. Not moving around while rotating should not introduce an error to ARCore because the initial position can be required through a matching of the unique points within the image.

Rotating the device manually for 2 minutes introduced positional tracking drift. An initial offset of 0.70 meters to the sides and 0.30 in length was recorded. During the test, the device was never moved more than 3 meters away from the initial position and was rotated with almost constant speed. Even though the initial position was lost, after pointing the camera towards the room plate that was used for the positioning, ARCore registered the tracking error and adjusted it automatically. This was most likely due to the higher density of feature points around the door.

Scenario 4: Keeping the device still for 10 minutes

This is a control test to test the error accumulation over time. Not moving for 10 minutes should not affect the AR tracking in any way whatsoever.

After scanning a room plate, the device was turned around and moved 2 meters away from the door and left alone for 10 minutes. When pointed at the room plate again, an offset of 1 meter to the sides and -0.9 meter in length was registered. Consequent tests without the rotation of the device have shown that keeping the camera pointed towards the room plate does not result in any drift whatsoever.

10.2 Testing the navigation logic

The navigation logic contains behavioral logic of the application when prompted with different destination and movement behavior of the users. The UI elements should always display the right information at the right time to not confuse them.

Scenario 1: Navigating from floor 0 to floor 3

Navigating between floors includes traversing stairs or elevators. These trigger special messages that explain the user where to move next, instead of showing them. The text of these messages and their timing have to be evaluated.

The test started with scanning a room plate from floor 0 and setting the destination to one in floor 3. Since all the stairs are relatively close to the rooms and the navigation would also point towards these stairs, it was hard to manually adjust the rotation of the device due to the short path that was displayed between the device and the start of the stairs. It was also clear that the NavMesh agent, a virtual representation of the user within Unity that moved parallel to the user, would get stuck in doors. This means that the navigation path would also get stuck at this point and the user had no way to know what went wrong. The only solution would be to rescan a room plate near a staircase and resume the navigation. Entering the staircase also required accurate pose information because the message to walk up the stairs would only trigger when entering the door towards them. This means that if the NavMesh agent got stuck close to the door, the message would also not trigger. However, the message did trigger when all conditions have been met and showed the correct explanation text.

Scenario 2: Navigating from the west end of floor 3 to the east end

This navigation path is the longest path within one floor. This path includes traveling over a bridge between the two ends, which clearly shows any offset in the position or rotation of the model.

This test also suffered from the previously mention tracking error of ARCore. When traveling over the bridge, the NavMesh agent would also get stuck on one side and only sometimes resume its tracking when arriving at the other. The resulting offset after successfully traversing the bridge between the two floor-halves resulted in an offset of approximately 10 meters to the side and 2.3m towards the length. The bigger offset towards the sides is most likely due to rotational tracking problems that came from the plate-scanning process. An offset of 0.5° can result in huge errors when traveling over a distance of over 100 meters.

11 Discussion and future works

This work provided a potential solution for an augmented reality indoor navigation system. Using ARCore and Unity to implement a software architecture which can be used to analyze ARCore and the impact of low tracking accuracy on such an indoor pedestrian navigation application.
The tracking has been split into two phases: User localization and continuous tracking. The former dealing with the problem of tracking absolute coordinates of users within the building. The latter to keep track of the user device's movement more frequently. In order to overlay digital information on the video feed, the exact motions of the device have to be tracked. ARCore uses these motions in combination with feature point detection methods to identify objects, including walls and floors, that can serve as an anchor for the AR elements.

The navigation system provided by NavMesh and the pose tracking system, which is a combination of ARCore's tracking and a text-based marker detection method, are modeled as modular systems that can be replaced by similar ones. A challenge was to include them into the model-view-presenter architecture chosen for this software. To also make the architecture as visible as possible, its structure has been implemented into the Unity scene hierarchy. This allows for developers who are new to this project to get a first impression of its functionality without the need to read any code. However, the limited structure of this hierarchy does not allow for a clean view into the connections between each software component. It only provides a general overview of them and what the model, view, or presenter are able to do.

The functional and non-functional specifications have almost all been met to a satisfying degree. The **model of the main building** has been completed and been **structured** in a way that new buildings can be added easily. **Scanning the room** plates also provides a reliable result. The text on them contains most of the necessary information and is completed by a database that contains their location within the model. **Tracking the pose** of a user within the building, however, is still a big issue. ARCore struggles with tracking complicated movements within a big environment and over a long period of time. To **project the results** of the tracking accuracy, using navigation elements, a line has been chosen. It displays the navigation path across the building. By logically comparing the accuracy of the direction of the line with the surrounding building and the position of the destination compared to the actual door, it is possible to also determine the degree of error that occurred during a navigation session. Additionally, due to the nature of the error accumulation rate, a time-based **accuracy display** has been added in the form of a change in line color. The last specifications of the software are related to its safety and its

improvements traditional navigation methods. The **safety aspect** was a deciding force when designing the stairs interaction. Instead of showing a 3D path along the stairs, the navigation gets interrupted by a message. This is to force the user to focus on the stairs and not look at their phone while using them. The **improvements of using the app** to help find a destination within the building are dampened by the low tracking accuracy and the resulting problems with NavMesh and Unity where the navigation information sometimes could get lost. If the tracking accuracy is too low, there is a possibility that the navigation elements could "get stuck" between doors or behind walls. This might be fixable by removing these doors, but the underlying issue of the misfunctioning tracking system still persists.

Acquiring the rotation of the device relative to the plate still poses problems that are negatively impacting the user experience. Alternative tracking approaches could improve on this. A combination between the current solution and a beacon-based system could provide more data about the user's movement within the building. By applying different filter techniques on the input of these system, their overall accuracy together can be greatly enhanced. Additionally, UI-focused obstruction strategies can be used to hide or limit the impact of tracking error accumulation. For example, telling the user what physical action to take next with turn-by-turn instructions or by using a virtual avatar that guides them to a general direction does not need a high tracking accuracy.
Developing a user experience focused application that assumes an unreliable tracking system could be a topic for future projects.

12 References

"5 Trends Appear on the Gartner Hype Cycle for Emerging Technologies, 2019." Accessed March 6, 2020. //www.gartner.com/smarterwithgartner/5-trends-appear-on-the-gartner-hype-cycle-for-emerging-technologies-2019/.

Atreyi Bose, and Chuan Heng Foh. "A Practical Path Loss Model for Indoor WiFi Positioning Enhancement." In *2007 6th International Conference on Information, Communications Signal Processing*, 1–5, 2007. https://doi.org/10.1109/ICICS.2007.4449717.

Azuma, Ronald T. "A Survey of Augmented Reality." *Presence: Teleoperators and Virtual Environments* 6, no. 4 (August 1, 1997): 355–85. https://doi.org/10.1162/pres.1997.6.4.355.

Bettadapura, V., I. Essa, and C. Pantofaru. "Egocentric Field-of-View Localization Using First-Person Point-of-View Devices." In *2015 IEEE Winter Conference on Applications of Computer Vision*, 626–33, 2015. https://doi.org/10.1109/WACV.2015.89.

Biswas, Joydeep, and Manuela Veloso. "WiFi Localization and Navigation for Autonomous Indoor Mobile Robots." In *2010 IEEE International Conference on Robotics and Automation*, 4379–84, 2010. https://doi.org/10.1109/ROBOT.2010.5509842.

Chengping Xu, and S.A. Velastin. "A Hough Transform with Integral Kalman Filter Refinement." In *IEE Colloquium on Hough Transforms*, 4/1-4/4, 1993.

Deng, Zhi-An, Zhiyu Qu, Changbo Hou, Weijian Si, and Chunjie Zhang. "WiFi Positioning Based on User Orientation Estimation and Smartphone Carrying Position Recognition." Research article. Wireless Communications and Mobile Computing, 2018. https://doi.org/10.1155/2018/5243893.

El-Sheimy, Naser, Haiying Hou, and Xiaoji Niu. "Analysis and Modeling of Inertial Sensors Using Allan Variance." *IEEE Transactions on Instrumentation and Measurement* 57, no. 1 (January 2008): 140–49. https://doi.org/10.1109/TIM.2007.908635.

Digital Trends. "FDA Approves AR Surgery Tool That Gives Surgeons 'X-Ray Vision,'" December 24, 2019. https://www.digitaltrends.com/cool-tech/xvision-spine-system-ar/.

Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst. "Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR." In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 193–202. Cambridge, UK: IEEE, 2008. https://doi.org/10.1109/ISMAR.2008.4637362.

Forlani, Gianfranco, Riccardo Roncella, and Carla Nardinocchi. "Where Is Photogrammetry Heading to? State of the Art and Trends." *Rendiconti Lincei* 26, no. 1 (June 1, 2015): 85–96. https://doi.org/10.1007/s12210-015-0381-x.

Google Developers. "Fundamental Concepts | ARCore." Accessed February 29, 2020. https://developers.google.com/ar/discover/concepts?hl=de.

Unity. "Get Real with Creating AR Games and Apps in Unity." Accessed March 5, 2020. https://unity3d.com/how-to/create-AR-games-in-Unity-efficiently.

Grubert, J., T. Langlotz, S. Zollmann, and H. Regenbrecht. "Towards Pervasive Augmented Reality: Context-Awareness in Augmented Reality." *IEEE Transactions on Visualization and Computer Graphics* 23, no. 6 (June 2017): 1706–24. https://doi.org/10.1109/TVCG.2016.2543720.

HALLAWAY, DREXEL, STEVEN FEINER, and TOBIAS HÖLLERER. "Bridging the Gaps: Hybrid Tracking for Adaptive Mobile Augmented Reality." *Applied Artificial Intelligence* 18, no. 6 (July 1, 2004): 477–500. https://doi.org/10.1080/08839510490462768.

Kim, Hee Jeong. Turn-by-turn navigation system and next direction guidance method using the same. United States US7844394B2, filed July 16, 2004, and issued November 30, 2010. https://patents.google.com/patent/US7844394B2/en.

Koo, Jahyoung, and Hojung Cha. “Localizing WiFi Access Points Using Signal Strength.” *IEEE Communications Letters* 15, no. 2 (February 2011): 187–89. https://doi.org/10.1109/LCOMM.2011.121410.101379.

Livingston, Mark A., Zhuming Ai, and Jonathan W. Decker. “Human Factors for Military Applications of Head-Worn Augmented Reality Displays.” In *Advances in Human Factors in Simulation and Modeling*, edited by Daniel N. Cassenti, 56–65. Advances in Intelligent Systems and Computing. Springer International Publishing, 2019.

MacIntyre, B., E.M. Coelho, and S.J. Julier. “Estimating and Adapting to Registration Errors in Augmented Reality Systems.” In *Proceedings IEEE Virtual Reality 2002*, 73–80, 2002. https://doi.org/10.1109/VR.2002.996507.

MacIntyre, B., and E. Machado Coelho. “Adapting to Dynamic Registration Errors Using Level of Error (LOE) Filtering.” In *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, 85–88, 2000. https://doi.org/10.1109/ISAR.2000.880927.

Mulloni, Alessandro, Hartmut Seichter, and Dieter Schmalstieg. “Handheld Augmented Reality Indoor Navigation with Activity-Based Instructions.” In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, 211–220. MobileHCI ’11. New York, NY, USA: ACM, 2011. https://doi.org/10.1145/2037373.2037406.

Nerurkar, Esha, Simon Lynen, and Sheng Zhao. System and method for concurrent odometry and mapping. United States US20170336511A1, filed May 15, 2017, and issued November 23, 2017. https://patents.google.com/patent/US20170336511A1/en.

Nix, Axel, and Andrew W. Gellatly. Turn-by-turn navigation system with enhanced turn icon. United States US7546207B2, filed May 21, 2004, and issued June 9, 2009. https://patents.google.com/patent/US7546207B2/en.

Pang, G.K.H., and H.H.S. Liu. “LED Location Beacon System Based on Processing of Digital Images.” *IEEE Transactions on Intelligent Transportation Systems* 2, no. 3 (September 2001): 135–50. https://doi.org/10.1109/6979.954547.

Pankratz, Frieder, Andreas Dippon, Tayfur Coskun, and Gudrun Klinker. “User Awareness of Tracking Uncertainties in AR Navigation Scenarios.” In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 285–86, 2013. https://doi.org/10.1109/ISMAR.2013.6671807.

Schmalstieg, Dieter, and Daniel Wagner. “Experiences with Handheld Augmented Reality.” In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 3–18, 2007. https://doi.org/10.1109/ISMAR.2007.4538819.

Android Developers. “Sensors Overview | Android-Entwickler.” Accessed February 25, 2020. https://developer.android.com/guide/topics/sensors/sensors_overview?hl=de.

Stoakley, Richard, Matthew J. Conway, and Randy Pausch. “Virtual Reality on a WIM: Interactive Worlds in Miniature.” In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 265–272. CHI ’95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995. https://doi.org/10.1145/223904.223938.

Sun, Yuxiang, Ming Liu, and Max Q.-H Meng. “WiFi Signal Strength-Based Robot Indoor Localization.” In *2014 IEEE International Conference on Information and Automation (ICIA)*, 250–56, 2014. https://doi.org/10.1109/ICInfA.2014.6932662.

Google Developers. “Supported Devices | ARCore.” Accessed April 21, 2019. https://developers.google.com/ar/discover/supported-devices.

Umek, Anton, Anton Kos, and Sao Tomaic. “Validation of Smartphone Gyroscopes for Angular Tracking in Biofeedback Applications.” In *2015 International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI)*, 294–98, 2015. https://doi.org/10.1109/IIKI.2015.70.

Unity. “Unity Public Relations Fact Page.” Accessed April 21, 2019. https://unity3d.com/public-relations.

Welch, G., and E. Foxlin. “Motion Tracking: No Silver Bullet, but a Respectable Arsenal.” *IEEE Computer Graphics and Applications* 22, no. 6 (November 2002): 24–38. https://doi.org/10.1109/MCG.2002.1046626.

Android Developers. “Wi-Fi Scanning Overview | Android-Entwickler.” Accessed March 2, 2020. https://developer.android.com/guide/topics/connectivity/wifi-scan?hl=de.

Yu, K., J. Chiu, M. Lee, and S. Chi. “A Mobile Application for an Ecological Campus Navigation System Using Augmented Reality.” In *2015 8th International Conference on Ubi-Media Computing (UMEDIA)*, 17–22, 2015. https://doi.org/10.1109/UMEDIA.2015.7297421.