TU WIEN Informatics

Opportunities of Augmented Reality Applications in Tunnel Construction

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software/Internet Computing

eingereicht von

Dominik Fenzl, B.Sc.
Matrikelnummer 01526544

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Mag.rer.soc.oec. Dr.rer.soc.oec. Christian Huemer
Mitwirkung: Dipl.-Ing. Marco Huymajer

Wien, 13. Dezember 2022

Dominik Fenzl	Christian Huemer

TU WIEN Informatics

Opportunities of Augmented Reality Applications in Tunnel Construction

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering/Internet Computing

by

Dominik Fenzl, B.Sc.
Registration Number 01526544

to the Faculty of Informatics

at the TU Wien

Advisor: Mag.rer.soc.oec. Dr.rer.soc.oec. Christian Huemer
Assistance: Dipl.-Ing. Marco Huymajer

Vienna, 13th December, 2022

Dominik Fenzl	Christian Huemer

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Dominik Fenzl, B.Sc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. Dezember 2022

Dominik Fenzl

Acknowledgements

I would like to express my deepest gratitude to professor Mag.rer.soc.oec. Dr.rer.soc.oec. Christian Huemer, who enabled me to work on this research topic and helped me with valuable feedback and all kinds of organizational tasks. This endeavor would not have been possible without Dipl.-Ing. Marco Huymajer, who helped me with every aspect of my thesis and was always eager to help me with every kind of trouble that arose. Additionally, I'm extremely grateful to Dipl.-Ing. Robert Wenighofer and Oleksandr Melnyk, MSc, who supported me with everything related to the tunnel visits at the ZAB and with valuable knowledge and expertise regarding tunnel construction. I would also like to express my deepest appreciation to professor Dipl.-Ing. Dr.techn. Gerald Goger for the financial support of my research.

Many thanks to Elisabeth Hauzinger, MSc, and Michael Mrazek for taking the time to test my prototypes and helping with any difficulties at the ZAB.

I also want to thank my friends Marco Fähndrich, Bernhard Ploder, and Alexander Hurbean, who I had the pleasure of studying with and who made the stressful days at the university bearable.

Thank you to my girlfriend, Marita Joy Kogelmann, for all her love and support throughout all the stressful times at the university. I would be remiss in not mentioning my family, who supported me in all kinds of ways throughout my entire studies. I want to especially thank my parents, Romana Fenzl and Thomas Fenzl, my siblings, Leonie Reiner and Bernhard Kautzky, and my grandfather Helmut Fenzl.

Kurzfassung

Der Tunnelbau gehört zu den letzten baubetrieblichen Bereichen, die immer noch stark auf papierbasierte Prozesse setzen. Die Erstellung papierbasierter Dokumentationen ist zeitaufwendig und fehleranfällig. Digitalisierung bietet die Möglichkeit diese Prozesse erheblich zu verbessern. Augmented Reality (AR) ist eine Technologie, die dieses Unterfangen unterstützen kann.

AR erlaubt Informationen zu bestimmten Positionen im Tunnel durch Lokalisierung automatisch bereitzustellen. Ein weiterer Vorteil von AR ist, virtuelle Informationen in einer realen Tunnelumgebung zu verorten. Durch die große Ausdehnung von Tunnelbauwerken sind andere Technologien zur Positionsbestimmung oftmals ineffizient. Trotz des enormen Nutzens von AR existiert wenig Forschung zu dieser Technologie im Tunnelbau. Dadurch ist es ist unklar, ob AR-Lösungen auf solche Prozesse anwendbar sind und wie die Lösungen implementiert werden können.

In dieser Arbeit wurden mithilfe eines Fokusgruppen-Interviews Problemstellungen im Tunnelbau identifiziert, die durch AR verbessert werden können. Anhand des Fokusgruppen-Interviews wurden zwei Problemstellungen ausgewählt, die durch den Einsatz von AR verbessert werden sollen. Zur Lösung dieser Problemstellungen wurden zwei High-Fidelity-Prototypen für die HoloLens 2 entwickelt, welche die Abnahmeinspektion und die Ortsbrustkartierung verbessern. Darüber hinaus wurde eine Lösung zur Lokalisierung im Tunnel entwickelt, die in beiden Protoypen zum Einsatz kommt. Usability-Tests geben qualitative Antworten über die Benutzbarkeit der entwickelten Prototypen.

Die Tests haben gezeigt, dass benutzbare AR-Lösungen im Tunnelbau mit aktueller Hardware realisierbar sind. Allerdings gibt es noch Probleme mit der Erkennung von Geometrie und Benutzereingaben der HoloLens, was den Einsatz solcher Lösungen in realer Umgebung noch nicht praktikabel macht. Des Weiteren hat diese Arbeit gezeigt, dass die Lokalisierung mit der HoloLens im Tunnel möglich ist. Es bedarf jedoch umfangreicherer Tests, welche die Resultate für reale Tunnelbauprojekte bestätigen.

Indem diese Arbeit Einblicke in die Nutzbarkeit gibt und Guidelines für die Implementierung solcher Anwendungen bereitstellt, dient sie als Startpunkt für zukünftige Forschung im Bereich AR im Tunnelbau.

Abstract

Tunnel construction is one of the remaining fields that still rely heavily on paper-based processes. Creating paper-based documentation is time-consuming and error-prone. Digitalization can considerably improve these processes. Augmented reality (AR) is one technology that could contribute to this endeavor.

AR can deliver information for specific positions in a tunnel through localization automatically. Another benefit of AR is locating virtual information in a real tunnel environment. Due to the large extent of tunnel structures, other technologies for position determination are often inefficient. Despite the enormous benefits of AR, there is little research on this technology in tunnel construction. Therefore, it is unclear whether AR solutions apply to such processes and how to implement them.

In this thesis, a focus group interview provided findings on problems in tunnel construction that are improvable with AR. Based on the focus group interview, we selected two problems to improve through AR. We developed two high-fidelity prototypes for the HoloLens 2 to solve these problems: acceptance inspection and tunnel face mapping. Additionally, we developed a solution for localization in the tunnel, which both prototypes utilize. Usability tests give qualitative answers about the usability of the developed prototypes.

The tests revealed that usable AR solutions are implementable with current hardware. However, some geometry and user input detection issues by the HoloLens still exist that make it not yet practical enough in real work environments. Furthermore, localization in tunnels with the HoloLens is possible. Regardless, more extensive tests are required to confirm the results for real tunnel construction projects.

This thesis serves as a starting point for future research on AR in tunnel construction by providing insights into usability and guidelines for implementing such solutions.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Tunnel construction, also known as tunneling, is an area that still heavily relies on paper-based processes for different operations [1]. Research projects such as TransIT aim to digitalize this area of construction [2]. This study is also part of the TransIT project and therefore seeks to contribute to the digitalization of the tunnel construction industry. The focus as a result of this lies in finding out if augmented reality (AR) has its place in improving tunneling processes. AR provides features such as detecting wall geometry. Furthermore, AR head-mounted devices (HMD) can show information on top of the environment's geometry. With such a device, users do not need to hold additional devices while working. Therefore, AR seems like a promising solution to solve problems in tunneling. Unfortunately, little research exists regarding AR in tunnel construction. The literature review for this study found only one paper from Zhou, Luo, and Yang [3]. Another critical point of this thesis is the localization inside a tunnel through AR. Currently, no easy or cheap solution exists that makes it possible for people working in tunnel construction to accurately find their position in real-time. This thesis tries to find out if AR can solve this problem in a reliable and usable way.

1.2 Problem Statement

The appearance of a tunnel is usually very homogeneous regarding its high gray walls with few distinctive features. This lack of distinctiveness presents a problem for AR algorithms since they usually rely on distinct features to localize themselves, for example, in rooms [4], [5]. Since the literature review did not find any research regarding AR in tunnel environments, this study must first clarify if such a localization works. Therefore, this study tries to provide answers to this question.

As already mentioned, localization inside a tunnel is currently not easily done. The Global Positioning System (GPS) is unavailable inside tunnels to determine the location. Installing Bluetooth beacons or other proximity sensors throughout a tunnel is also not feasible since tunnels are usually very long. AR solutions could solve this issue by utilizing the sensors on a device to locate itself in a tunnel by recognizing certain features.

As mentioned, the tunneling industry often relies on traditional or paper-based processes. Documentation is an example that heavily relies on paper-based textual descriptions and two-dimensional drawings. Information stored on paper-based reports includes geological and hydrological characteristics of the surroundings, performed activities, and applied material [1]. Various processes create this information, such as inspections, tunnel work, and tunnel face mappings. It is desirable to retrieve all information corresponding to the specific location in a tunnel. However, this is time-consuming to achieve because of the large number of documents. An AR solution that can localize itself could retrieve this information if stored in a digital format, such as in TIMS [6]. Furthermore, AR can aid the process of capturing information with visualizations according to positions in a tunnel. Additionally, AR HMDs already include cameras that users could use to capture photos for documentation and inspection purposes. Automatically adding positional information for specific points in a tunnel is also quickly done if the localization works. For example, users could virtually highlight points on the wall to save the position of support measures. Nevertheless, the usability of AR solutions must be tested before employing them in real tunnel projects. Bad usability could otherwise hinder the acceptance of such solutions.

1.3 Structure of the Thesis

The remainder of this thesis is structured as follows. Chapter 2 explains the methods utilized to answer the research questions formulated. Afterward, Chapter 3 introduces tunnel construction. This introduction includes the essential processes for constructing a tunnel and a summary of the state-of-the-art approaches found in tunneling and general construction. Chapter 4 deals with the introduction to AR. This chapter includes a comparison with virtual reality and an explanation of essential algorithms and techniques. Furthermore, this chapter introduces state-of-the-art solutions utilizing AR in construction and tunneling. Chapter 5 explains problems in tunneling, improvable with AR found from a focus group interview. This chapter also includes requirements for prototypes to solve selected problems. Chapter 6 explains the devices, tools, and approaches existing to implement AR applications. This explanation includes comparisons of different AR frameworks, devices, and localization approaches. Afterward, Chapter 7 explains the implemented prototypes. This chapter explains the implementation of the localization, acceptance inspection, and tunnel face mapping prototype. Chapter 8 shows the collected information from the usability tests and a discussion about the usability of AR in tunnel construction. This chapter also includes guidelines for future AR development that target tunnel construction. Lastly, Chapter 9 concludes this thesis by highlighting the information gained from this study and providing recommendations for future research.

1.4 Research Questions and Expected Outcome

The following list displays the research questions for this thesis.

1. RQ1: How practical is localization in tunnels with the currently available augmented reality hardware?
2. RQ2: What problems in tunnel construction can be improved through the adoption of augmented reality?
3. RQ3: How helpful are AR solutions in tunnel construction?

The expected outcome of this thesis is to answer these research questions. Additionally, this study aims to provide a baseline for future research regarding AR in tunnel construction. Therefore, this thesis will provide insights into the usability of AR applications in tunnels and guidelines for designing them.

CHAPTER 2

Methodology

This chapter deals with the scientific methods used in this study to answer the previously mentioned research questions. First, we explain the design science research approach and the methods used in each part of this framework. This part also includes the justifications for each method and why we chose them for this thesis. The last section deals with the limitations of this study to show where future research is necessary to strengthen the found results.

2.1 Design Science Research

Design Science Research of Hevner, March, Park, *et al.* [7] is the basic framework that combines the different methods used in this thesis. One aspect of this framework is the delegation to future research for incremental improvements of the findings. Since not many previous works exist as a foundation, we chose the Design Science Research framework to guide future research in developing AR solutions for tunneling.

We created the knowledge base in the rigor cycle, according to Hevner, March, Park, *et al.* [7], through the summary of foundational literature. This knowledge base includes information about the basics of AR and the actors and processes involved in tunnel construction. We employed the guidelines of Kitchenham and Charters [8] in a scope appropriate for a master thesis to summarize the state-of-the-art AR solutions in construction and tunneling. The literature review involved the following steps. First, we identified studies appropriate for this topic. We achieved this with the following search strategy.

1. Augmented reality in construction search keywords.
    - AR Construction
    - Augmented Reality Building

- Augmented Reality BIM
- Augmented Reality Construction
- AR BIM
- Augmented Reality Construction Operation

2. Augmented reality in tunneling search keywords.

- augmented reality tunneling
- AR tunneling
- AR BIM tunneling
- AR tunnel construction
- Augmented reality tunnel construction

We used the previous search keywords in Google Scholar and IEEE Explore. Each mentioned search keyword implicitly contains a logical *AND* operator instead of the spaces between the words for the search on the mentioned platforms. Additionally, we also refined the keywords by skimming through the search results. With this approach, we found that *AR tunneling* did not yield many results related to tunnel construction. Since AR also stands for argon, we found more papers related to chemistry than augmented reality. Searching for *Augmented reality tunneling* led to the only paper found that is concerned with AR in tunnel construction and fits the criteria. Lastly, it can be mentioned that the term *tunneling* also describes a phenomenon in AR that is not connected to tunnel construction. We often found this term when searching for tunneling in combination with AR.

After collecting the papers, we further filtered them with the following selection criteria.

1. The studies should not be older than five years.
2. They should provide an explanation of how their solutions were achieved, including the technology and hardware used.

After that, we skimmed through the remaining papers and searched for keywords containing *AR*, *augmented reality*, *tunneling*, and *tunnel construction*. Only if a combination of those keywords was present the paper was selected. For the state-of-the-art AR solutions in construction papers search, the approach was the same. However, the search only included the word *construction* combined with variations of the term *augmented reality*. After that, we read the entire paper.

We summarize the findings from the literature review and the foundational literature in the chapters Current State of Digitalization in Tunnel Construction, Basics of Augmented Reality and its Place in the Construction Industry, and Tools & Approaches Applicable to the Prototypes. The chapter Requirements also includes findings from the AR papers

and the foundational literature for the design of AR user interfaces with usability in mind.

We conducted interviews with experts from the civil engineering/construction domain to determine the actual needs of the environment to adhere to the relevance cycle of Hevner, March, Park, *et al.* [7]. The experts selected were a convenience sample of academics from the Montan University Leoben and the TU Wien.

First, we conducted a focus group interview according to the guidelines of Krueger and Casey [9]. We chose this method because input from experts from this domain is needed. The other reason is that the literature did not provide any evident problems AR could solve. A focus group interview helps to identify ideas from a broader point of view and can highlight problems relevant to the environment through an open discussion between multiple experts. This interview was performed with a group of five people and aimed to identify possible solutions in tunnel construction. This interview focused on using AR in tunnel construction and operation phases. We chose these phases since the first research question deals with the localization inside tunnels, and in these phases, the workers work at the site. The questions asked in the interview were the following.

1. Which problems exist in tunnel construction (construction/maintenance and operation phase)? We focused on problems that can utilize the following methods.
   - Visual indication (i.e., in the form of holograms)
   - Spatial understanding
   - Localization inside the tunnel
2. How could solutions look like for the previously found problems?

The focus group interview was conducted online via the online conferencing tool Zoom[1] and contained the following steps.

1. The HoloLens 2 and AR was briefly explained in the form of a presentation to give the participants a rough idea of the capabilities of this technology.
2. After the presentation, the participants were invited to collaborate on an open-source online whiteboard called Excalidraw[2]. This whiteboard already contained the question for the participants to discuss. We drew inspiration for this approach from Moore, Mckee, and McCoughlin [10], who also conducted an online focus group interview and utilized whiteboard software. They found that the whiteboard was a valuable tool to direct the flow of the focus group to new topics or return to already introduced ones. The whiteboard helped them stay focused on the critical research questions. However, this only worked for some groups of them. Chat as the only

[1] https://zoom.us/
[2] https://excalidraw.com/

communication part could be a source for some of the problems they experienced. This focus group interview, therefore, also uses voice and video as communication. We then asked the participants to write their ideas on the whiteboard in keywords and discuss them afterward. For the moderation style, we used the guidelines of Krueger and Casey [9]. Tips from this included asking participants further questions. Another tip was asking other participants for approval of previous answers and clarifying inconsistent comments. Since the participants were already writing down their thoughts, we reduced the detailed note-taking to a minimum. We then enhanced the written-down information and clarified it on the whiteboard during the interview. After the interview, we exported the whiteboard as an image. We show this image in Figure 2.1.

3. Lastly, the collected information was summarized and categorized by the problems identified.

Following the focus group interview, we conducted an additional semi-structured interview to refine and gather additional requirements for possible solutions we selected as prototypes. We held this interview via the online conferencing tool Zoom. For this interview, we used the collected guidelines from Kallio, Pietilä, Johnson, *et al.* [11]. This method was chosen to answer specific questions about the found solutions but also to have the possibility to gather more specific or additional information from the participants during the interview. We gathered more detailed information through additional questions during the interview. After that, we had to narrow down the problems during the interview by asking for more clarification about the implementations. For the list of questions, we used the previous knowledge collected in the knowledge base and the initial interview. We formulated these questions with words like „who“, „what“, „when“, and „where“ to increase the chance of getting descriptive answers according to [11]. To remove any ambiguities and check in which order the questions seem most appropriate, we also tested the questions internally [11]. We list the questions created to identify the requirements for selected solutions in the following.

- Questions for the localization inside the tunnel
  1. Are there some indications for distance inside the tunnel? If yes, what indications exist, and how precise are they?
  2. Is there internet usually available in the tunnels?
  3. How would you design an optimal solution for localization inside a tunnel?
  4. What is the maximum acceptable time to set up the markers for a distance of 100 m?
  5. How many markers are acceptable to place per ten meters?
- Acceptance Inspection Prototype
  1. Who is involved?

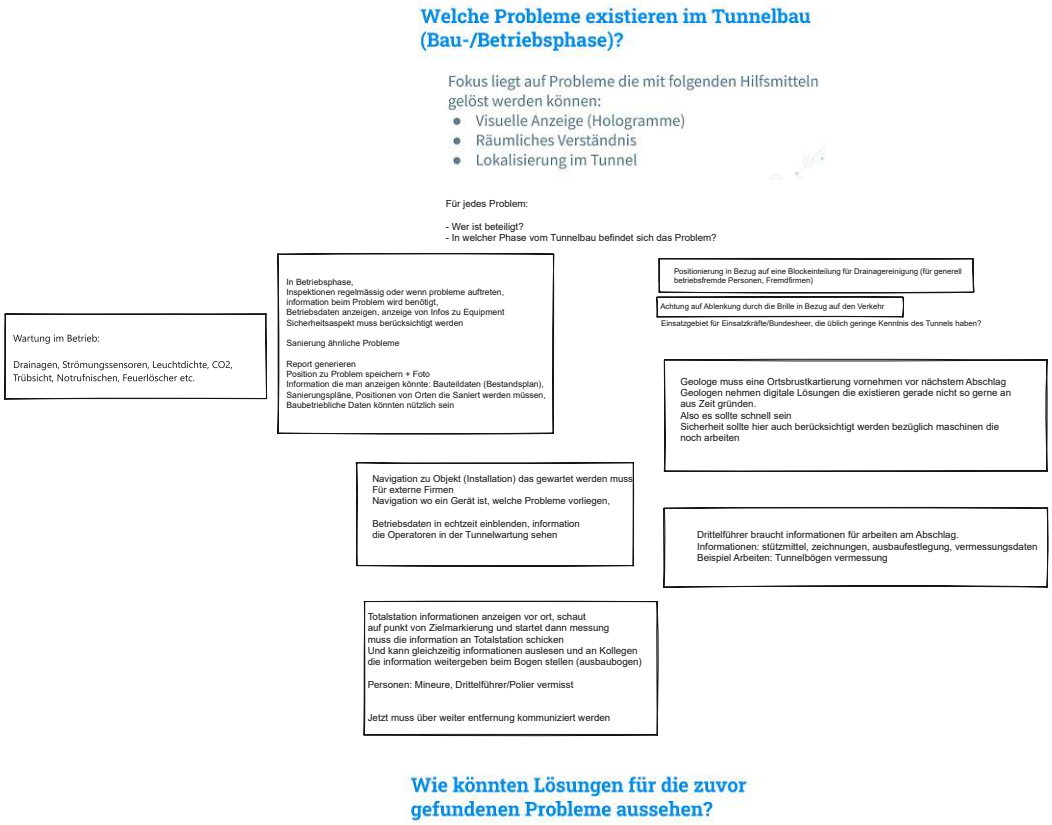

Figure 2.1: Result of the focus group interview to identify problems and solutions in tunneling utilizing AR.

2. What are the usual procedures for such an inspection?
3. What information should be displayed?
4. How should the opening action of the information look like? (voice command, floating button, fixed button for a location)
5. What interaction possibilities should the user have with the information? (e.g., a floating window showing information, should the window be movable, should it be fixed at a location)
6. How should the user be able to interact with the window? (e.g., scaling, increasing font size, changing text color, searching for text)
7. What are some example problems?
8. Should the user be able to add information? If yes, what information?
9. What precision is needed for the localization of the round?

- Tunnel Face Mapping Prototype

1. What is a geologist's procedure during the tunnel face inspection?
2. What kind of information does a geologist have to note down?
3. Is there a fixed list of observations the geologist has to note down for each tunnel face? If yes, what are these observations, and what does the information look like?
4. How does the inspection of a geologist look like? Is this done quickly? Are there things measured in between?
5. How does an example of a sketch of a tunnel face look like?
6. Should a tunnel face note be editable?
7. What happens with the tunnel face notes after they are created? Are they needed in future phases?
8. Who uses the information of the tunnel face notes? Are these people using the information for planning purposes or at the site itself again?
9. Should there be a photo of the tunnel face without the sketches and text?

- Round Task Information Prototype

1. Does a foreman also work on previous finished rounds?
2. What tasks does a foreman have to do?
3. When is this problem relevant?
4. What kind of information has to be displayed?
5. Should the menu be interactive? For instance, the user clicks through different steps or checks what was already done.

6. What interaction possibilities should the user have with the information? (e.g., a floating window showing info, should the window be movable, should it be fixed at a location)
7. What precision is needed for the localization of the round?

We then refined the gathered requirements in the form of user stories. Chapter 5 includes a summary of these user stories and more detailed information about the problems and ideas collected from these interviews.

We created artifacts according to Hevner, March, Park, *et al.* [7] to answer the research questions. Guidelines and approaches from the researched knowledge base, as well as the gathered inputs from the interviewed experts, were used to implement these artifacts in the form of high-fidelity prototypes.

With the prototypes, we wanted to answer whether localization inside tunnels using AR is feasible. For this, we incrementally improved and tested the prototypes since we had to investigate the uncertainties first. Examples of these uncertainties are if and under which conditions the detection of the spatial awareness system works because environments with less distinct features are harder to detect by the AR algorithms used. At first, we tested the prototypes at a subway station with a long section of feature-poor walls for initial insights. Later multiple tunnel visits at Zentrum am Berg[3] (ZAB) were scheduled to conduct field tests. These field tests further increase the relation to the environment to ensure the prototypes work in such locations. Multiple approaches were tested out, namely utilizing markers and markerless anchors.

In these field tests, we also took measurements to ensure reasonable accuracy. Figure 2.2 shows this measuring approach. We drew multiple lines with a distance of one meter between each line. Afterward, we walked over the markings and compared the distance displayed from the solution with the markings on the ground. We then inspected if the solution showed the correct tunnel chainage once we overstepped a line. We documented the results of these tests and prototype changes in Chapter 7.

We also used field tests to test the prototype implementations. Employees at the tunnel site provided helpful information to increase the usability of the user interfaces. Furthermore, We tested if the solutions work in the tunnel as expected regarding the visibility conditions and the similar-looking environment. The tests also led to new requirements and design choices for prototype improvements. We documented the resulting design changes and insights in Chapter 7.

We evaluated the developed artifacts through qualitative usability tests at the ZAB. These tests should answer the question of how usable AR applications are in tunnels. We used the guidelines of Barnum [12] to conduct the usability tests. For the usability tests of the prototypes at the ZAB, we used a convenience sample of 5 participants. We prepared multiple scenarios with tasks for each prototype. The participants had to go

[3] https://www.zab.at/

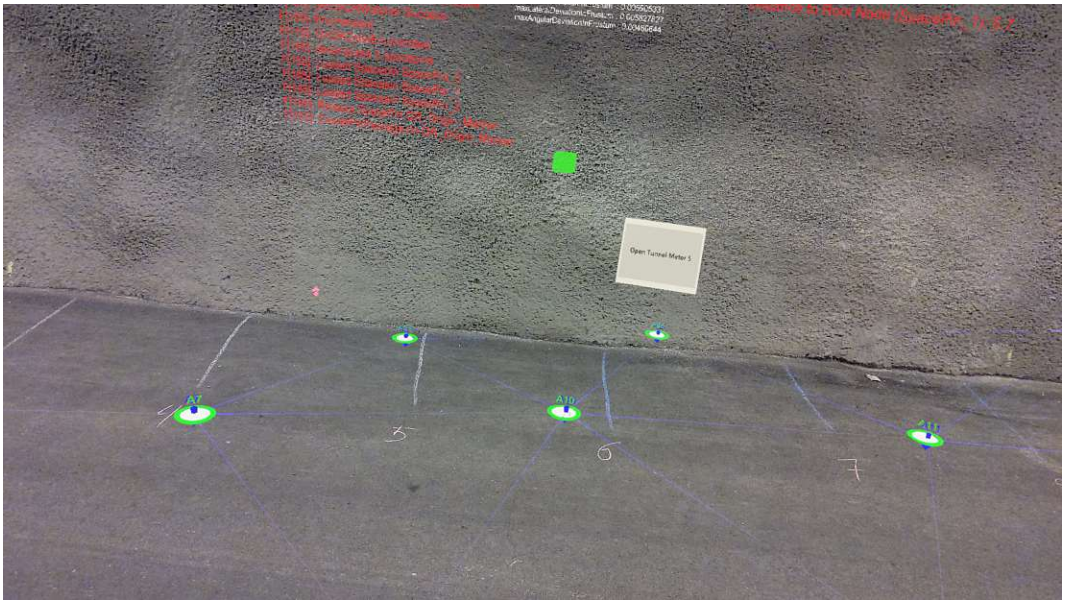

Figure 2.2: The measurement of localization inside the tunnel to check if the position of a tunnel chainage is correctly identified for the user's position. The photo was captured by a HoloLens 2.

through these tasks to test and interact with all aspects of the prototypes. Before the participants played out the scenarios, we showed them how to interact with the HoloLens. We then asked them to complete the HoloLens tutorial provided with the device to familiarize themselves with the controls. In the following, we will explain these scenarios.

Before the participants tested the prototypes, the first step was to set up the localization by placing markers over a distance of 30 m. The participants achieved this by placing the markers on three tunnel chainage signs, each 10 m apart.

The following scenario was created for the participants to go through to test the usability of the acceptance inspection prototype.

1. The participants first had to start a new inspection by switching to the user mode.
2. After that, they were asked to go through all the rounds visualized through the borders. While traversing the borders, the users should look at the information window, which holds the information for the current round.
3. If the round information window contained no support measure (e.g., shotcrete), they should use the *include in report* button to mark the round as inspected.
4. If the current round contained anchors, the participants were asked to use the mark place functionality to mark spots on the tunnel lining to record the number of anchors. The participants would mark anchors in the real world, but the tunnel

we could use did not contain any visible anchors. For this step, we asked the participants to mark three places on the tunnel lining. So if the users found three anchors, they marked two on the left lining and one on the right lining. After the marking, the participant had to take a photo of the marked spot.

5. We then asked the participants to write a note for each found marking. The participants had to annotate the anchor. So, for example, if one round had three anchors, then three spots were marked with the notes *Anchor1*, *Anchor2*, and *Anchor3*.

6. Lastly, the participants had to end the inspection by previewing the report. This step opens a window that includes the summary of the found spots and locations. The summary had to be checked by the participants. If the participants did not find any issues, they had to press the *save* button to conclude this scenario.

For the tunnel face mapping prototype, we created the following scenario.

1. First, the participants added a marker right at the tunnel face, where they previously had placed the markers. This information is later needed to assign the tunnel face mapping to the most recent round.

2. After that, the participants had to place a canvas, which has the form of a rectangle, over the tunnel face. For this, the participants had to scale the canvas to cover the whole tunnel face and position it in front of it.

3. Subsequently, the participants had to use different tools to annotate the tunnel face (free drawing, adding text, drawing polygons). We show an example of this mapping in Figure 2.3.

4. Lastly, the participants were asked to create a photo of the created tunnel face mapping with the appropriate button. This step shows the participants the photo captured, which they had to accept with a confirm button to complete this scenario.

We asked the participants to think aloud while solving the tasks at hand during the usability test. Observations of the participants were also logged by filling in notes on a prepared set of questions. We also observed the participant's views during the tests through a tablet that streamed the live video footage. These questions are the following and were filled out for each prototype scenario separately.

1. Does the user seem confident, i.e., does the user complete the tasks at a fast pace, or are there problems that make the user hesitant?

2. How long did the user take for each task? (The tasks correspond to the previously mentioned steps. We measured the time for each task.)

    - Setup

Figure 2.3: An example of a tunnel face mapping captured during the usability test.

- Task 1
- Task 2
- And more tasks, depending on how many tasks are available for each scenario.

3. How skilled is the user with hand gestures? (for placing objects, text, or drawings)
4. Does the participant struggle with the virtual keyboard?
5. Does the participant have questions about how to open the menus?
6. Is it clear for the participant how to traverse the menus for the given tasks? Or do they seem to search for the different menus available?
7. Are there problems with the solution? For example, are virtual objects or menus drifting out of the participant's range?

After the test, each participant had to fill out a survey of open-ended questions to get insights into how they perceived the used prototypes, if they would change anything and if they would want to use such solutions in the future. We show these questions in the following.

1. How intuitive was the interaction with the HoloLens for you? Did you struggle to get used to it, or was it easy for you?

2. Which types of buttons were easier to use for you? The buttons that had to be clicked from far away or near you (e.g., from the hand menu or dialogues)? Can you explain if you struggled with a certain type of button?

3. Could you imagine wearing the HoloLens longer?

4. Were the windows located in a good position to read and interact? Would you prefer to have the windows nearer or further away? Maybe larger windows? Please explain if you would have liked anything to be different.

5. Was the hand menu or the floating window more practical for you? Would you have liked any menu option on a different window, i.e., one option switched from the floating window to the hand menu, or was the mix of both menus good, in your opinion?

6. How easy was the setup process for the localization for you? Did you have any problems? (The process where you had to mark the tunnel signs.)

7. Could you imagine using this process for longer distances? Or do you think there could be any problems with this approach?

8. Was the white grid displayed on the walls irritating for you to place the markers, or was it helpful?

9. How intuitive was the process of completing the tasks with the acceptance inspection prototype for you? Did you struggle with any particular part of the implementation, for instance, with the user interface?

10. Was it clear how to interact with the user interface to complete the next step of the scenario? Or would you have liked a clearer indication of what to do next?

11. What did you think of the window containing the information for the specific round? Was it clear enough to get the information needed? Was there something hard or not possible to read?

12. Would you use such a solution instead of a paper-based solution? Or do you see any problems with this approach? (Can also be minor problems with the implementation that you would like to have changed.)

13. How intuitive was the process of placing the canvas over the tunnel face? Did you have any problems with the scaling or moving part?

14. How was the interaction with the canvas for you? Was it easy to use the available tools to annotate and sketch on the canvas?

15. Was the hand menu intuitive, or did you encounter any problems using it?

16. What do you think of the photo capturing and annotating of the tunnel face? Was it easy to capture it all into one image, or would you like hints on achieving a better result?

The gathered requirements and the result from the evaluated prototypes are additions to the knowledge base. They should help researchers in this area to improve on the solutions and implement new AR solutions for the discovered problems. Additionally, this developed knowledge base answers the research questions from this thesis to give future researchers a basic framework to understand what is implementable with AR in this environment with the current technology and which are the current challenges.

2.2 Limitations

The convenience sample of participants in the usability tests led to initial insights into the applicability of AR solutions in tunnels. However, a larger sample size of participants is needed to gather better insights into the usability of the proposed solutions. Additionally, it is also necessary to test them with persons that work in the respective field for which the prototypes should help. Future usability tests should therefore include geologists and personnel from the site supervision. Nonetheless, the current results serve as first impressions of the usability of the developed prototypes since the tasks executed were simplified for the tests. Tests, including longer distances inside tunnels, are necessary to check if the localization works continuously. For instance, to check if wormholes occur on longer distances. A wormhole is an effect where the detection of similar features leads the device to think of being at another location. Microsoft[4] mentions that this phenomenon can occur. However, the provided option to use QR codes as markers could solve this problem. Rescanning the QR code when walking to the next tunnel chainage sign could achieve this. This procedure should realign the coordinate system of the solution correctly again. Lastly, the tests also revealed that participants need more time to interact with the HoloLens confidently. Therefore, multiple tests with the same participants would be beneficial to gain insights into how long the participants need to work confidently with the HoloLens.

[4] https://learn.microsoft.com/en-us/HoloLens/HoloLens-environment-considerations

CHAPTER 3

Current State of Digitalization in Tunnel Construction

This chapter deals with the current practices in tunneling and state-of-the-art digitalization solutions in the construction industry. At first, the tunneling process is explained in detail to understand the activities, personnel, and resources needed to build a tunnel. After that, we define the current practices in the construction industry. We then show the state-of-the-art solutions and practices which utilize digital technology. These practices and solutions include modern approaches that are used in structural engineering as well as in mechanized tunneling. Lastly, TIMS [6] is explained in detail since this is an essential basis for the prototypes created during this thesis.

3.1 Tunneling

Hemphill [13] describes tunneling as the discipline of digging a hole under the surface of the earth in contrast to mining, which is concerned with retrieving valuable materials from underground. The book *Practical Tunnel Construction* [13] gives a comprehensive insight into tunnel construction and is the basis for the following paragraphs.

Tunnel construction is an essential discipline in the field of civil engineering. According to Hemphill [13] the first use cases for constructed tunnels were for water and warfare. Nowadays, we use tunnels for various applications. Muir Wood [14] presents a concise overview of tunnel usage, and we summarize this in the following.

One example is overcoming physical obstacles, such as bodies of water or mountains, that hinder the construction of transportation (road, rail). Another example is the usage in urban planning, economy, and security, e.g., subsurface transportation, sewerage and sewerage treatment, and cold stores. Lastly, tunnels are also an excellent structure to control a specific environment concerning temperature, humidity, and vibration. These

controlled environments are necessary for applications such as high-precision industries and laboratories.

Despite tunneling having such a simple definition, its construction is a challenging operation. People working in tunneling must consider any factors in the different stages of tunnel construction. For example, the planning phase already decides how to excavate and build the tunnel. These decisions heavily depend on the types of rock and soil present. During construction, miners have to deal with uncertainties, such as movements, new types of rocks, and water leakage. Geologic and hydrologic conditions are generally the most critical information for tunnel planning, engineering, and construction [13].

In a tunnel construction project, there are usually many different actors involved. Kvasina [1] created a concise overview of these different actors in a conventional tunneling project. The following provides a summary of this overview to better understand the scope of such an undertaking.

1. A *client* is a natural or legal person that hires a contractor to fulfill certain services. They normally have a representative called construction management at the construction site. The construction site management represents the interests of the client.

2. A *contractor* enters into a contract with a client to perform certain services.

3. The job of the *site supervision* is to monitor and inspect the services demanded by the contractor. This assignment also includes checking deliveries, personnel involved in the project, compliance, and meeting deadlines. Lastly, the construction site management also handles the final inspections and examination of all protocols and invoices.

4. *Local geologists* are responsible for creating geological reports and the documentation of the tunnel face. The tunnel face is the area that is currently being worked on to drive the tunnel forward. After each tunnel advance, the geologists have to record the tunnel face and note down information about the lithology, mountain water ratio, and other aspects.

5. The assignment of the *project manager* of a contractor is to ensure that the construction project is successful from the economic and technical standpoint.

6. The *construction manager* is responsible for leading and supervising the construction site. This manager reports directly to the construction company and represents the site to the client. An important task of the construction manager is the organization and coordination of the construction. Compliance with costs, laws, and deadlines is also the area of responsibility of this manager. Lastly, different types of documentation have to be created by this person, such as the builder's diary and the recording of problems.

7. *Site managers* are subordinates of the construction manager. They have two main tasks. First, they must take care of the work preparation, such as making sure the right building plans are available where needed. The overall work progress is also closely monitored by them. Second, the site manager is the first contact of the site foreman for any problems that arise during construction. Such problems can be geological or with the used machines. The site managers generally work very closely with the site foreman to manage the team of construction workers.

8. *Shift engineers* support the construction manager and management of the execution. They are mainly responsible for parts of the construction documentation. Some examples of this documentation include daily construction reports, decade plans, and field measurement sheets. Additionally, shift engineers make target/actual performance comparisons of material and construction time and document data for later recalculations.

9. The *site foreman* takes care of the observation, organization, and documentation of the construction tasks and the usage of heavy equipment and personnel. This position sits between the construction manager (or site manager) and the industrial workers.

10. A *foreman* leads a team of construction workers and completes tasks provided by the site foreman and manager. This position is a direct subordinate of the site foreman. An essential job of the foreman during the tasks is ensuring the work progress and safety.

11. *Miners* work on the forefront of the tunnel advance. They perform all the tasks during an excavation cycle, such as operating heavy equipment and installing support measures.

12. A *construction administrator* carries out all operational and commercial topics of the construction site agendas. Some examples of the activities for this position are recording the costs fully and divided into the correct periods. Additionally, the construction administrator also checks the project's success using a profit and loss account and verifying and posting invoices.

13. The *back office* supports the project manager in the management of the construction site. A focus lies here on the technical and economic aspects of the project.

Just as in the lifecycle of other structural facilities, the building of tunnels can be divided into five phases. We summarize four of those phases from Kochendörfer, Liebchen, and Viering [15] below. The fifth, operations and maintenance phase, is summarized from Bergeson, Ernst, *et al.* [16].

1. *Project development* is the phase concerned with everything from the project idea until the investment decision to realize the project. This phase also includes risk analysis and examinations concerning economics.

2. The *conception* phase is needed to answer the questions about functional connections, constructive structure, requirements for further planning, and technical building systems.

3. In the *planning* phase, there are three important planning stages. Firstly, the design planning stage exists to create a collection of drawing documents considering the constructed tunnel's technical, structural, biological, ecological, economical, and energy aspects. Secondly, this phase needs approval planning that works intertwined with the design plan. This phase includes creating, collecting, and submitting the documents required for the construction's approval. Moreover lastly, during the execution planning, an execution-ready solution is constructed from the results of the previously mentioned plans.

4. In the *realization/construction* phase the actual construction is instructed. Previously created documents and plans help to accomplish the construction. However, deviations from previous plans can happen because of newly created plans during construction and unforeseen problems. Documentation is critical in such scenarios. A significant period in this phase is the project closure. This period is essential for transitioning from the realization to the operations and maintenance phase. One aspect of this is acceptance, which compares the commissioned tasks with the actual result. Created documents and onsite inspections during the previous phases exist to aid this acceptance process. Additionally, this period includes tasks such as consultation regarding the briefing of personnel, maintenance, and operation contracts. Another critical point is archiving construction files such as the organization and project book. Finally, this phase concludes with the coordination transmission from the object supervisor to the builder or user.

5. As the name suggests *operations and maintenance* consists of two parts. The operations part is concerned with the safety and efficient use of the tunnel. Furthermore, this includes evaluating gases (oxygen, carbon monoxide) and proper luminance. Moreover, checking the functional systems, for example, ventilation, pumping, and lighting. Additionally, monitoring the traffic and incidents inside tunnels is essential in the case of road tunnels. The maintenance part is more interested in the tunnel structure's safety, preventive and corrective work, and proper equipment servicing. Processes in maintenance include flushing drains, washing tunnel structures, and changing light bulbs. Verifications and measurements also have to be made. Corrective measurements in case of equipment failure or other needs for repairs also need to be made in this phase. Lastly, tunnel system rehabilitation is also an essential point for this stage. This point comprises large-scale repair programs of individual tunnel systems. For instance, to fix structural problems or replace the lighting system.

This thesis focuses on the realization and the operations and maintenance phases. For this reason, we explain these phases in more detail in the following.

3.1.1 Realization Phase

As already mentioned, there are multiple methods to construct a tunnel. In the following, we will explain a few standard techniques presented by [13].

The most frequently used technique for tunnel construction is called **conventional tunneling** and is also the focus of this thesis. *Drilling and blasting* is one of the most popular methods in conventional tunneling, but various other means also exist. This approach first drills holes into the tunnel face with exact positions, diameters, and depths. The drilled holes are then filled with explosives to cause fractures in the rock formation. Loose rock removal from the crown is the next step once the site is ventilated and clear of dust and gases generated from the explosion. Support installation happens after this step. After that, tunnel workers load and transport the blasted rock (muck) out of the tunnel. According to Hemphill [13], the last two steps can be reversed depending on the ground. The transportation of the muck can happen before the support installation if the ground is stable enough so that no ground support is needed.

The advantages of this method are the lower capital costs compared to other techniques. This technique also provides flexibility in terms of the tunnel shape and direction. Machines cannot construct all types of tunnel shapes and directions. This method is also very versatile since it works on many rock types. The most significant disadvantage of the drill and blast method is the considerably lower advance rates. The method also poses an increased safety risk to personnel due to the blasting. One of those issues is the risk of overbreak due to arising vibrations.

Mechanized Tunneling is a method utilizing machines that became very popular in the last two decades [17]. The machines used in this method are called Tunnel Boring Machines (TBM). Using a TBM at a specific construction site requires adapting and changing the machines according to the prevailing geological properties. Therefore, planners must consider the type of rock and soil while selecting the TBM. Otherwise, the machine may be over-stressed or even break down.

Figure 3.1 depicts a Tunnel Boring machine. In contrast to the traditional drilling and blasting method, tunnel boring machines use torque and thrust to break rocks and drive forward. The forward thrust is provided by pushing the cutter head against the tunnel face with hydraulic thrust cylinders (3) from the precast segmental lining. Grippers (2) provide grip for this forward thrust. They are parts that get extended and pushed against the sidewalls. The cutterhead (1) is the front part of a TBM. Different cutting tools are attached to the cutterhead, depending on the type of rock. The correct cutting tool then helps with the rock penetration for the tunnel. Lastly, a conveyor collects the muck from the bottom of the cutter head. This muck is transported behind the tunnel boring machine and carried away.

Hemphill [13] mentions that the advantages of mechanized tunneling include increased work safety due to the lack of need for explosives compared to drilling and blasting. Vibrations are therefore also no concern, and fewer over breaks happen. Lastly, TBMs can have higher advance rates with fewer deployments of personnel. The disadvantages

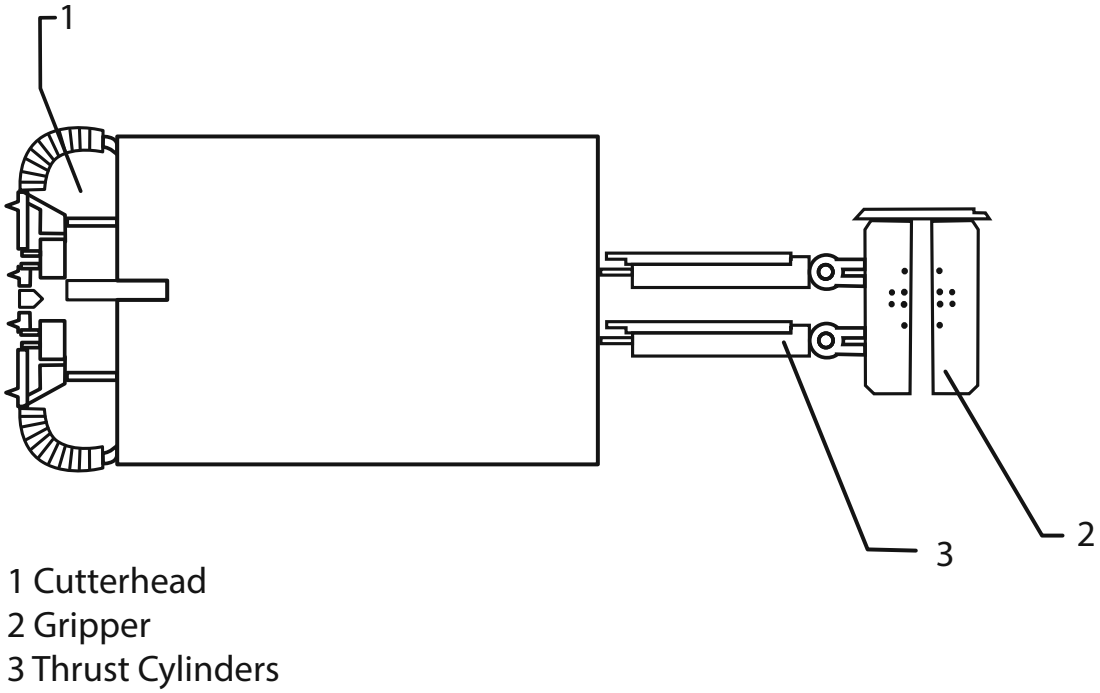

Figure 3.1: A minimalist sketch of a TBM highlighting important components (Inspired by [13]).

of this method are the higher capital cost. Also, the time to mobilize TBMs is very long. Personnel has to have a certain level of skill to use such machines. Lastly, these machines are not very adaptable to changes in types of geology and limited reuse opportunities. However, reuse is a problem designers have improved by selecting tunnel sizes available TBMs can construct.

3.1.2 Operations and Maintenance

As explained previously, the operations and maintenance phases consist of two parts. Both parts happen in parallel and are equally essential to ensure a safe and efficient tunnel. Another part that is not directly assignable to either operations or maintenance is inspection. Since this is also very important in this phase, we will explain it in the following. We will also explain operations and maintenance to gain insight into the tasks and people involved.

Inspection

Another critical aspect in the operations and maintenance phase is inspecting the tunnel in various areas (e.g., structure and equipment). We, therefore, summarized some common tasks during inspections from [16] in the following.

- Maintaining tunnel inventories and records.
- Reporting found safety or structural problems.
- Maintaining current load ratings on all tunnel structures.

Operations

To highlight how much effort is needed to operate a tunnel, a summary of the different tunnel personnel and their respective tasks for an exemplary highway tunnel is presented in the following, summarized from [16].

1. The *tunnel manager* is responsible for the tunnel facility. Tasks for this position include financial management during the operation of a tunnel. Records of the tunnel facility also need to be managed by the manager. Moreover, a program must be created and operated that complies with the relevant regulations, policies, and laws.
2. A *facility engineer* is mainly concerned with the maintenance and further improvements of the tunnel. This engineer performs various evaluation tasks to check previous maintenance records and if repairs or improvements are necessary.
3. *Tunnel supervisors* take care of the day-to-day tunnel operations and communicate arising issues to the tunnel manager and facility engineer. They schedule the maintenance and repair work and other work assignments. Additionally, they also respond to incidents and handle lane and traffic closures.
4. The *tunnel operators* are responsible for monitoring everything that happens inside a tunnel. This monitoring includes traffic and sensor measurements such as air contamination, lighting intensity, power consumption, and water accumulation. They also activate the emergency systems in case of major incidents.
5. *Tunnel foremen* usually lead a small team of specialists in a certain area and sometimes general laborers. The tunnel foreman is specialized personnel utilized by the tunnel facility and is concerned with a specialized area of practice in tunnel operations.
6. *Tunnel mechanical specialists* are responsible for tasks concerning mechanical equipment. Examples of equipment they are responsible for include pumps and ducting to air conditioning units. Tasks that have to be undertaken by tunnel mechanical specialists include oil replacements, blade cleaning, and other routine maintenance.
7. A *tunnel electrical specialist* takes care of the electrical tasks inside a tunnel, such as working on the electronic drive system and power distribution. They also complete tasks such as changing batteries, running generators, and checking

fire detection/suppression equipment. Lastly, the tunnel electrical specialist also handles routine tasks and repairs of electrical nature.

8. The *tunnel electronics specialists* handle all communication, low voltage power equipment, support equipment and systems. For instance, fire alarm systems, HVAC control systems, lane control signals, and CCTV systems all belong to the workspace of a tunnel electronics specialist.

9. *Tunnel laborers* take care of various tasks during the tunnel operation. They wash structures, clean drains, install light bulbs, and are responsible for general housekeeping. As already mentioned, they also support specialists in their work if required.

10. The job of a *safety officer* is to coordinate the response in case of an emergency by communicating with the different emergency services. This officer also conducts training and drills to prepare for emergencies.

11. *Security officers* work between the tunnel facility and various police units. They also respond to emergencies, inspect payloads of vehicles, and escort hazardous freight in the tunnel.

We see from the previously mentioned personnel for the operations of a tunnel that a tunnel needs many undertakings to keep it efficient and safe.

Maintenance

Maintenance is an important topic during the operation of a tunnel and is fundamental to the safety and integrity of a tunnel. A comprehensive and practical source for the tunnel maintenance is [16]. Therefore, we use this as the base material for the following paragraphs.

A program is needed to provide a minimum level of maintenance conduction for the usage during tunnel operation. Such maintenance programs must be optimized to provide public safety, reduced cost, and an acceptable level of service. Many forms of documentation, information, and knowledge about the tunnel are needed to create these programs. This collection of knowledge must include everything from the start of the life of a tunnel (i.e., the construction). Information about constructed sections and geological properties collected early on helps with this. Documentation is also necessary during this phase, as problems that arise during various phases of a tunnel can help identify underlying problems and detect them earlier if they reappear.

There are multiple reasons why so much effort is needed to keep a tunnel safe and operational. One reason is groundwater, which introduces moisture into tunnels, contributing to faster corrosion and deterioration. Common degradation processes related to groundwater are spalling of concrete because of steel reinforcement corrosion. Corrosion can also cause short-circuiting and other electric failures. These processes can also worsen

the protective finishes and coatings. Moreover, clogging of drainage pipes can also occur. Lastly, this degradation can lead to section loss and reduced element strength.

We can divide a maintenance program into three categories. These categories are *preventive maintenance*, *corrective maintenance*, and *tunnel system rehabilitation*. While most of the focus of a program should be on preventive maintenance. Bergeson, Ernst, *et al.* [16] suggest that 70 % to 80 % of activities under a preventive maintenance measure are good practices.

Preventive maintenance is vital to increase the life of a tunnel as well as reduce the probability of failure. According to [16] multiple maintenance approaches should be used in this category. There is the cyclical method that is done periodically at determined time intervals. Conditional maintenance uses measurements and observation to plan measures. Furthermore, lastly, there is the predictive-based method that uses statistical analysis and model forecasting to predict future failure. Some everyday tasks for this category are regularly removing debris, snow, and ice to provide safety. Another vital task is washing the tunnel structures. Washing these tunnel structures is needed to make the inspection work more manageable when the interior becomes dull with dirt. Therefore periodic tunnel washing is needed, depending on the tunnel type and environment. Cleaning drainage inlets and pipes should be incorporated with tunnel washing to prevent clogging.

Corrective maintenance, also called on-demand maintenance, is essential in the case of problems that are difficult to predict. Such problems can be caused by vehicle damage, defective equipment, or unexpected tunnel system failure. So this category is needed to deal with these problems by developing emergency plans. Typical tasks for this category include dealing with concrete detachment. Concrete can detach for various reasons, like deterioration, corrosion of rebar, and faulty placement techniques. Therefore one task is to remove detached and loose concrete from the ground, walls, and ceilings. Another problem is explosions, earthquakes, floods, or rock slides which can cause general damage to the tunnel. A damage inspection typically follows such a scenario. Lastly, handling groundwater seepage through liners is also tasked to mitigate long-term corrosion and deterioration.

Tunnel system rehabilitation is usually the process of a large, cost and engineering-intensive repair project. Such a repair project deals with tunnel systems near their expected end of life. Repair jobs in this category include replacing the lighting system, overhauling the ventilation system, and completing considerable structural repairs.

3.1.3 Traditional Project Management and Documentation

The previous sections clearly show that many different actors are working on a tunnel, from design to operation and maintenance. Furthermore, many different tasks have to be handled by these persons, which makes the organization a challenging effort. This great organizational effort makes it more concerning that most of the project management in conventional tunneling relies on traditional methods [18]. Moreover, documentation needed for further operations and to gain insights into the current status is still produced

in paper form [1]. Paper documents are highly inefficient in tunneling, as mentioned by Huymajer, Operta, Mazak-Huemer, *et al.* [6]. The aggregation of this significant amount of information could be helpful. Invoicing periods could use this information about excavation and support measures. Various overviews could also make use of this information. Such overviews could be created by querying the data for different information (e.g., finding all activities done during water penetration). Such an overview could also provide easier target/actual comparisons of the undertaken activities. However, this is highly time-consuming to achieve through laborious manual searches through paper-based documents.

Sometimes the reports and documents created are also digitized, i.e., scanned and stored as a PDF file. However, further automatic processing is usually more challenging since manual tasks that tend to introduce errors are still needed for this to work [6]. Additionally, scanned documents make it harder to edit or add information and often result in rescanning of printed-out documents [19]. This process introduces multiple versions, and comparing the file history can take time.

Digital files are also sometimes used for documentation and management. For example, these files can be in the form of spreadsheet files, 2D CAD (computer-aided design) drawings with the design information and photographs and images [6], [18]. Even so, interpreting all files and having a cohesive overview is only possible with a proper data integration or management solution. This loose file structure also makes it challenging to manage organization processes [6].

The previous paragraphs show much room for improvements in conventional tunneling with modernized approaches. Digitalization is a critical factor that could contribute to this effort by providing better management, easier overviews, and automatic processing of collected information in all life cycles of a tunnel.

3.2 Digitalization of the Construction Industry

While conventional tunneling lags behind in terms of digitalization and state-of-the-art project management, we can usually find a more modern approach in the general construction industry and mechanized tunneling. To get an overview of the digitalization methods used in construction, we explain them in this section in more detail. First, we explain the so-called common data environment (CDE) since this topic is essential to understand how all the information is stored and used in other techniques. After that, we continue with an introduction to building information modeling (BIM) in construction. Lastly, the usage of BIM in tunneling is discussed, giving insights into the special adaptions required to fulfill the unique requirements of tunnel construction.

3.2.1 Common Data Environment

A construction project produces many different pieces of information because of the many actors involved and the different phases it goes through. Therefore, a well-defined

and organized management strategy is needed to accommodate this complexity of data. One such option is the so-called Common Data Environment (CDE), a collaborative digital project space according to Preidel, Borrmann, Exner, *et al.* [20]. This space allows project members to access the data they need. The definition of different states and how they can change is also possible with this space.

Preidel, Borrmann, Exner, *et al.* [20] provide a comprehensive explanation for CDE, which is used in the following paragraphs to summarize this topic. A CDE is a centralized space to manage, check, share and collect different information. This space allows project members to enter the information in their specific domain and make it available for other domains. The goal of a CDE is to create a platform to exchange information while keeping the information consistent. Therefore, the project members must adhere to the definitions of specific techniques and proceedings. For example, for every state change, specific quality assurance proceedings are executed for every data element needed. Common data environments, therefore, lead to a clearly defined type of cooperation with all project participants.

A CDE can reduce the risk of redundant data and high availability through centralized information management. Additionally, CDEs make it easy to reuse information and serve as a centralized archive for documentation that needs to be stored indefinitely.

3.2.2 Building Information Modeling

The DIN EN ISO 19650, 2018/2019 standard [21] describes BIM (Building Information Modeling) as the „*use of a shared digital representation of a built asset to facilitate design, construction and operation processes to form a reliable basis for decisions*“. Although multiple standards describe BIM, the previously mentioned one is the only one accepted worldwide and internationally standardized according to Spengler and Peter [22].

Spengler and Peter [22] describe Building Information Modeling more practically as a cooperative way to work with all project members on a shared building model, ideally with a collective data environment. This method enables collaboratively storing, improving, and digitally providing different kinds of information and using it as a cooperation instrument.

The critical point is that BIM is not software but a method that needs software solutions to work. Therefore, multiple software solutions are available for different aspects of a built asset. It is, therefore, important that the different solutions can exchange data with each other to still have a coherent view. We summarized examples of different usable BIM software solutions (summarized from [22]).

- CAD 2D and 3D can store plans and models of buildings. This information can include annotations and be converted to other file formats to include more information than typical CAD files.
- Software can be used in statics for simulation and calculation purposes with the given project model.

- The whole lifecycle can contain multiple different cost and procedure plans.
- Financial accounting can be made easier for the project by, for example, storing receipts and invoices digitally.
- Information about the constructed building can be used in project management to link different information and give a more detailed view of the progress and stages.
- Software solutions can enhance documentation management and is an essential part. Such solutions can keep a digital version of the various documents and notes throughout the whole lifecycle of a project. This information collection makes documentation searchable, editable, and digitally evaluable (i.e., statistics and analysis)
- Contract management can be automatized by, e.g., automatically scanning documents and parsing the included information. Furthermore, creating digital approvals and other workflow automation can aid the contract process.

Another important aspect of BIM is the overlapping view of all lifecycle phases in a construction project, according to Stange [23]. These phases are the idea-finding phase, the planning phase, the construction phase, the operations and maintenance phase, and the dismantling and recycling phase. Documentation and information are generated and digitally stored during the previously mentioned phases. Each subsequent phase can then use this information available in databases. Therefore, the BIM method leads to a systematic way of working that affects how information is collected and data is maintained.

The SWOT (Strengths, Weaknesses, Opportunities, Threats) analysis of Spengler and Peter [22] gives a concise overview of the advantages of the BIM approach. The author highlights the advantages of the different lifecycle phases and the whole lifecycle. We summarize these advantages in the following.

1. There are multiple benefits of BIM over the whole lifecycle, such as the increased cooperation that leads to higher efficiency, reduces redundancy for information and work (e.g., for modeling or planning), and higher transparency between the project members. An overview of all lifecycle phases also eases the creation of a lifecycle cost analysis. Additionally, BIM can increase the cost and deadline precision through quicker quantity determination and the creation of service descriptions. Lastly, BIM leads to higher usage of computer-aided solutions, making future digitalization efforts more approachable.
2. In the planning phase, BIM can help with clash detection to detect problems in partial models that different members in their respective expertise create. Moreover, the different information that BIM makes available such as the geometry and location of a project, can be used in simulations and computational calculations.

This information is used, for example, in statics and building physics, and immensely reduces the cost of the handover.

3. For the construction phase, the building model makes offer considerations from building companies easier through more straightforward expense determination and a more accurate invoice. The model can also simulate the construction process to find problems early on.

4. BIM also brings improvements to the operations and maintenance phase. For example, facility managers can use the digital model with all its information for room sizes and connections for outlets and building services. This model can also compare operations and maintenance costs with the planning and building costs. Furthermore, simulations through scenarios can provide optimizations of the costs over the lifecycle of a building.

3.2.3 BIM in Tunneling

While BIM started primarily for the usage of structural engineering, it recently also found its way into tunnel construction according to Flora, Fröch, and Gächter [24]. Sharafat, Khan, Latif, *et al.* [18] mention that BIM has its own term in the tunnel industry and is called TIM (Tunnel Information Modeling).

The principal usage of BIM in tunneling is the same as in structural engineering. However, Flora, Fröch, and Gächter [24] mention that a model of the building ground is an essential aspect of tunnel construction not found in typical structural engineering. It is essential to capture specific attributes, such as the soil condition and the type of rock present. These attributes are necessary, e.g., for forecasts. BIM solutions should also digitally integrate the construction site. For instance, mapping the excavation processes can increase the efficiency in larger projects where multiple tunnel works happen in parallel [18], [24].

As stated by [24], modeling the building model for a tunnel project is more manageable than for structural engineering projects. Fewer components and less coordination between professional planners are the reason for this. Nevertheless, more complex scenarios require digitally captured safety and maintenance concepts. Moreover, tunnels also have a higher lifespan (in the range of hundreds of years). Therefore, such long-lasting projects need to consider the maintenance of this data.

The Brenner Base Tunnel already shows the advantages of using BIM in tunneling. The usage of this method led to improved design optimization, productivity, and data sharing and made the decision-making process easier [18].

3.3 TIMS

TIMS (Tunneling Information Management System) created by Huymajer, Operta, Mazak-Huemer, *et al.* [6] is also a solution to help with the digitalization of conventional tunneling projects. As the name suggests, this solution fits into the tunneling information

modeling approach. This tunneling information management system is also vital for the prototypes developed in this thesis since it serves as the backend containing all the information used or added. This section explains TIMS in more detail and how it is relevant to this thesis.

The goal of TIMS is to digitalize the documentation process in tunneling using the new Austrian tunneling method (which falls under the category of the conventional method). As already mentioned, conventional tunneling usually relies on paper-based documentation, which can lead to errors and is also time-consuming. TIMS tries to solve these issues.

The authors of [6] found three documents present in most construction sites that are still mainly created as paper-based documents. These documents are excavation and support sheets, construction progress diagrams, and daily construction reports. Tunnel personnel must handle the previously mentioned documents daily, i.e., preparation, signage, and submission. Furthermore, the documents are signed, scanned, and sent to the responsible construction company after the responsible checks and confirms the correctness. The documents are also important for creating various statistics and reports, such as the following.

- *Tunneling statistics* aid in the target/actual comparisons and measuring the tunnel construction performance. These statistics also help identifying the time in which the tunneling occurred to determine pay/time-related costs.
- *Excavation and support statistics* capture the excavation and support measures used during invoicing.
- *Field measurement sheets* confirm the amount and performance of the construction work.

Huymajer, Operta, Mazak-Huemer, *et al.* [6] created a conceptual tunnel model to accommodate the documentation process during the tunnel construction. A class diagram represents this model. Figure 3.2 displays this class diagram. This model captures various aspects currently present in paper-based documentation of the tunneling in the realization phase. As seen in Figure 3.2, the classes capture information such as rounds, related activities, and support definitions. Moreover, other relevant classes, like Personnel and Products, represent the materials used.

As seen in Figure 3.3, the architecture consists of a backend and two frontends.

The backend was implemented based on the open-source Enterprise-Resource-Planning (ERP) system and framework Tryton[1]. Tryton already comes with functionalities such as access control, a view engine, a report engine, and a workflow engine. These features make the ERP framework a sound basis for utilizing the information from the documentation,

[1] https://www.tryton.org/

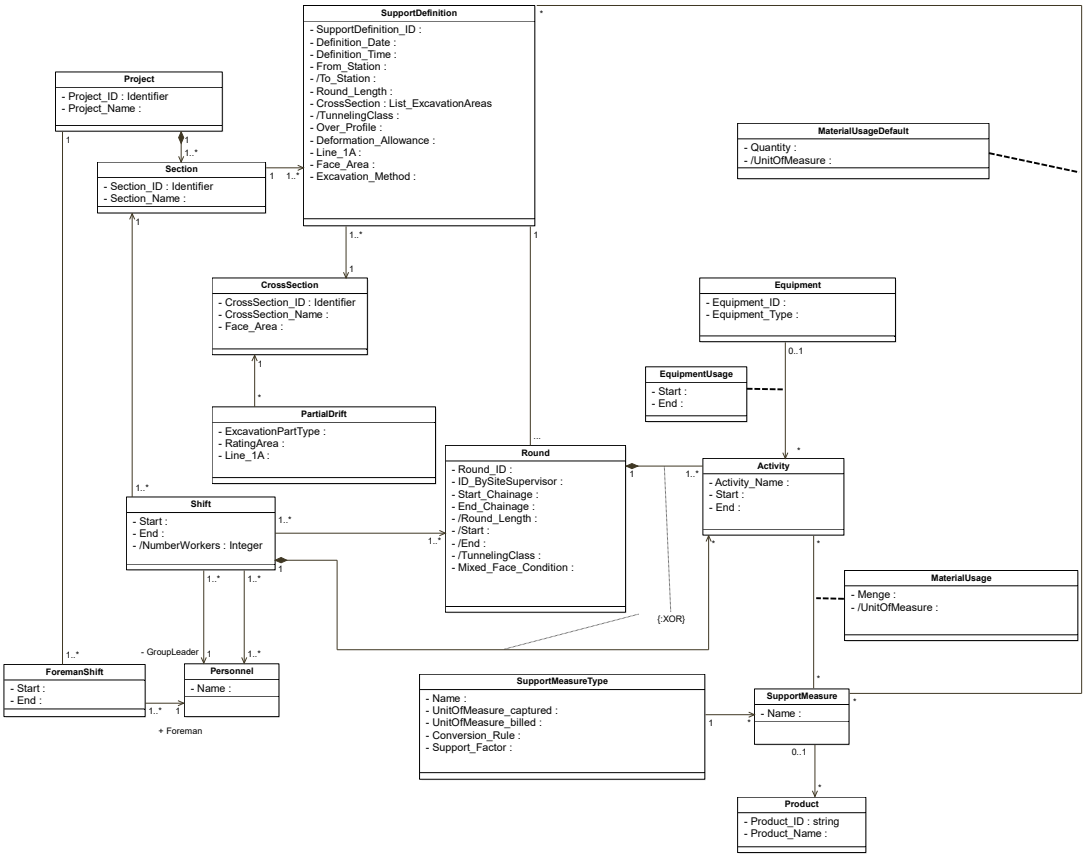

Figure 3.2: The conceptual tunnel model as a UML class diagram [6].

i.e., the tunnel models previously mentioned. Additional definitions of workflows, data constraints, and access rules make this solution even more usable. A Remote Procedure Call (RPC) and a Representational State Transfer (REST) API is available through the backend to communicate with the two frontends and other applications that need to communicate with this solution in the future.

A front end for mobile use and one for desktop use enable the interaction of project members to achieve a digital documentation process. The mobile-focused solution was created as a simple web application to allow easy data entry at the tunnel face. This front end focuses on recording activities and material usage for a tunnel round. Foremen can select their current shift and add information such as the shift personnel and the current tunnel round. Additionally, the foremen can use a timetable diagram to add, edit or delete activities.

The intent of the second front end is more of an administration view with additional functionalities. In this application, users can freely edit data in the available views, and it is possible to set up new tunneling projects. It is also possible to gain an overview of the

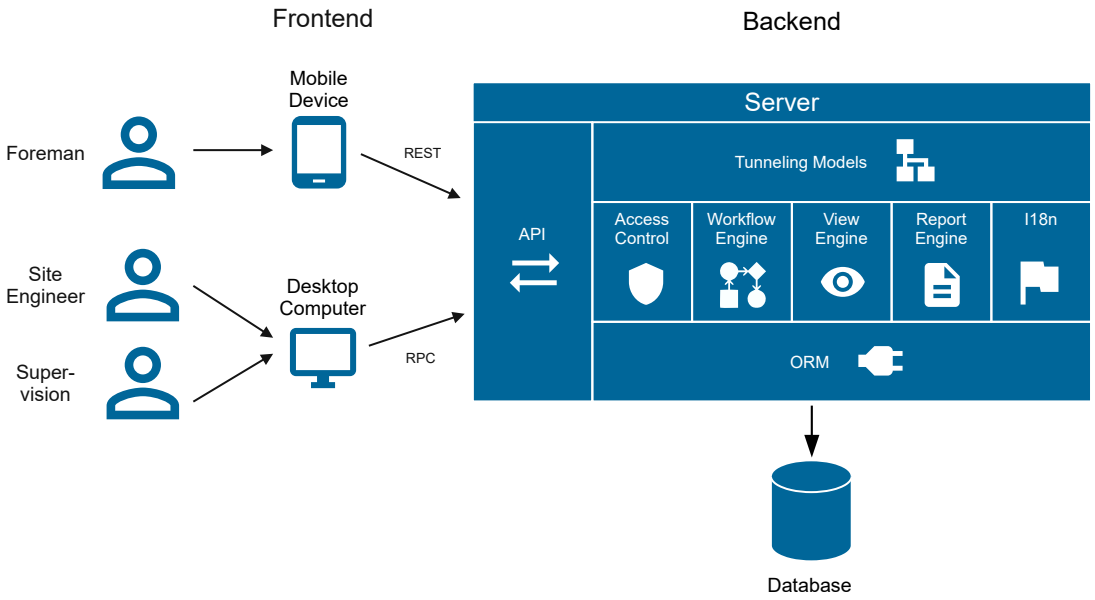

Figure 3.3: The architecture of TIMS [6].

whole tunneling process with this application, e.g., the activities done and materials used. Lastly, as already mentioned, other digital documents can also be created automatically from the incorporated information. The creation of tunneling and support statistics is also possible with this.

TIMS provides significant advantages to a tunnel project compared to the traditional paper-based documentation process. It helps create a centralized vision of such a project in all tunneling phases and mitigates the creation of redundant and erroneous data.

CHAPTER 4

Basics of Augmented Reality and its Place in the Construction Industry

This chapter concerns the explanation of a basic understanding of the technology behind AR and its applications in construction. The first part introduces AR, how it works, and how it compares to similar technology. Thereafter, we give an insight into AR trends in the construction sector, including some examples of current research in this area. Lastly, we present state-of-the-art AR applications in tunnel construction.

4.1 Augmented Reality

This section is concerned with the fundamentals of AR. Therefore, AR is explained more coarsely, including its definition, technology, and usage. Afterward, we provide a detailed clarification of the SLAM algorithm and why it is essential for AR solutions. We continue by describing Virtual Reality and Mixed Reality and what differentiates them from AR. Lastly, we discuss AR's current limitations to consider them during the development of the prototypes for this thesis.

4.1.1 Introduction

Kipper and Rampolla [25] describe AR as an overlay of digital or computer-generated information, e.g., images, audio, video, and touch, over a real-time environment. This information with AR enables the composition or superimposition of virtual objects with the real world, which the users can see. Lastly, the authors mention that AR does not entirely replace reality but supplements it.

According to Kipper and Rampolla [25] AR has three distinct characteristics that need to be present. Firstly, AR needs to blend virtual and actual information. Secondly, one can interact with AR in real time. Furthermore, AR must be used and operated in a 3D environment. AR is not one single technology; instead, it is a combination of multiple technologies. This combination can bring digital information into visual perception.

There are multiple devices currently available to use AR. These devices all have in common that they have a camera and a way to display digital information. We summarize examples of commonly used AR devices from [25].

- *Personal computers with webcams* can utilize markers that users hold into the camera view. This detection of markers by the computer allows users to see the augmentation on the computer screen while interacting with the marker. Such applications are also sometimes used for advertisements with QR codes.

- *Kiosks, digital signage, and window displays* are stations where customers can scan items to find out more information about them. AR can enhance these machines also. An example is a Lego Store kiosk, which displays the built Lego set that resides inside the box by holding it into the camera. Customers can also rotate the box to see the 3D model from all angles.

- *Smartphones and tablets* are pervasive devices used for AR simply because of their availability and portability. Such devices can scan different kinds of markers to display 3D augmentations on the screens. Other sensors, such as inertial sensors, compasses, and GPS, can further enhance the stability and selection of information displayed. Smartphones and tablets can also use a markerless approach to augment what is visible on the camera. Most AR frameworks for mobile devices support this feature, for instance, ARCore[1] and ARKit[2]. With this capability, rooms can be scanned, for example, to place 3D models on surfaces.

- *AR glasses* are the most sophisticated devices for AR applications since they solely focus on AR with their integrated sensors and hardware. AR glasses have the same capabilities as mobile devices regarding AR applications, but with increased performance and additional sensors (e.g., Magic Leap 2[3], HoloLens 2[4]). What also makes these devices stand out against the previously mentioned ones is that the display is positioned directly in front of the eyes. Therefore, no device has to be held by the users. Instead, they can look around the environment by moving their heads and see augmented visuals.

[1] https://developers.google.com/ar/develop
[2] https://developer.apple.com/documentation/arkit/#overview
[3] https://www.magicleap.com/device
[4] https://docs.microsoft.com/en-us/HoloLens/HoloLens2-hardware

4.1.2 SLAM

As mentioned, AR applications can use markers to visualize 3D elements on them or in their vicinity. However, another technique also makes it possible to augment larger scenes (e.g., a room) without markers. This technique is called markerless and is available in most modern AR applications. The algorithm for this technique is called Simultaneous Localization and Mapping (SLAM). Most AR frameworks (e.g., ARKit, ARCore, Vuforia[5], Microsoft's Mixed Reality Toolkit[6]) use SLAM.

SLAM is a solution developed for the area of robotics. According to Cadena, Carlone, Carrillo, *et al.* [26], SLAM initially solved issues for mobile robots with no knowledge of their work environment, i.e., no map. GPS can also be a solution for navigating unfamiliar terrain. However, SLAM solves the problem of GPS being unavailable indoors. With success in researching robotics, it is now also being used extensively in AR.

As stated before, SLAM is crucial for AR. Therefore, in the following paragraphs, we summarize the excellent introduction to SLAM from Jakl [27].

A system that uses SLAM consists of the following four parts.

1. The system has to collect data from various *sensors* included in mobile devices and AR glasses. Examples of such sensors are the camera, gyroscope, and accelerometer. Additionally, sensors that can enhance accuracy are light sensors, GPS, and depth sensors.

2. *Front-End* is the initial process of detecting important points with feature extraction and tracking them for SLAM. Feature extraction is explained more in detail in another blog post of Jakl [4]. In this, he explains that a feature point (extracted feature) is a distinctive location in an image that can be detected reliably. This point can also be detected when things such as the rotation, camera angle, blur, or lighting change. These feature points must also be associated with 3D positions via tracking in a video stream. We will give a more detailed explanation of this process later on.

3. The *backend* is responsible for creating the relationships between different frames in a video stream, localizing the camera's position, and the overall geometrical reconstruction of the environment. There are different types of reconstruction methods. For example, the sparse reconstruction uses the environment's key points and whole 3D point clouds.

4. *SLAM estimate* is the aggregation of the tracked features, the relations and locations of these features, and the positioning of the camera within the environment.

[5] https://developer.vuforia.com/

[6] https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05

SLAM Implementation

For SLAM, there is not one single implementation but lots of different ones for each AR framework. We, therefore, summarize the explanation from Jakl [4], [27] of SLAM implementations to get a deeper understanding of how SLAM works in general in the following paragraphs.

As already mentioned, the first step after acquiring the environment's image data is to find feature points. A separate algorithm finds such locations in images. A good candidate, according to Jakl [27], is the *BRISK (Binary Robust Invariant Scalable Keypoints)* algorithm by Leutenegger, Chli, and Siegwart [28] because it is fast and efficient enough for mobile devices.

The BRISK algorithm is based on Rosten and Drummond's [29] FAST algorithm. FAST is a corner detection algorithm often used as a basis for feature detection. FAST works by looking at the neighbors of each pixel p in a circle. When a certain threshold of connected pixels has a higher or lower brightness than p, the algorithm marks it as a corner. BRISK uses a minimum of nine consecutive pixels within a circle with a radius of 16 pixels. Moreover, the neighbors must be adequately higher or lower for each central pixel for the algorithm to recognize it as a feature. The algorithm also downsizes the images to reduce inaccuracy due to scale.

Another crucial step in the feature detection process is the key point description. This description defines that all found key points must be unique fingerprints so they can be found again in different images. For example, images can have other lightning or a different perspective. BRISK uses for this descriptor a binary string that is 512 bits long. This string concatenates the result of the brightness comparisons between different samples that are around the analyzed center point. Lastly, the algorithm calculates characteristic directions to ensure that rotations do not interfere with the detection. The method to find these directions works by finding out the angle of a cut through a key point area that results in the most significant difference in brightness between each side of the cut.

Jakl [27] explains further steps based on the ORB-SLAM implementation of Mur-Artal and Tardós [30]. ORB-SLAM uses a similar algorithm to BRISK for finding key points, called ORB, created by Rublee, Rabaud, Konolige, *et al.* [31]. This implementation analyzes each frame for these key points and then stores them in a map with references to keyframes where they have been detected. This map is critical to finding these key points in future frames and improving previous points' accuracy.

The next important step in SLAM is to map the previously found key points from 2D images to a 3D coordinate. Jakl [27] calls these coordinates *map points*. ORB-SLAM takes these key points per frame and matches them with the previous camera frames. The algorithm can estimate the initial camera pose in real time with this information. Afterward, the algorithm projects the map into the new camera frame to find additional (previously found) key points. The solution further refines the camera pose if it finds other key points. Lastly, with triangulation, new map points are created with the matched

key points in connected frames. The algorithm triangulates the map points with the 2D positions and the rotation and translation between the frames.

Loop detection and closing is another process needed in SLAM. This step checks if key points found in a frame match previously visited locations. In this process, ORB-SLAM checks if the similarity surpasses a set threshold to know if the location is already known. The algorithm then uses this information to correct the coordinates across all available location information for the current location, and the map is updated.

Unfortunately, the companies behind AR frameworks such as ARCore, ARKit, or the Mixed Reality Toolkit (MRTK) have not publicly stated how they implemented their SLAM algorithms. However, Jakl [27] mentions that all the implementations include the information of inertial measurement sensors in addition to the camera.

Challenges

Jakl [4] highlights problems with the SLAM algorithm in the ARCore feature detection visualization demo. Namely, smooth surfaces such as white walls with few features are hard to detect and not detected at all with the ARCore implementation. However, the concrete pillar found in the video shows many features found by ARCore.

Anchors

Finally, the most popular AR frameworks use SLAM as an anchor or spatial anchor feature. These anchors bind 3-dimensional objects or other information to a fixed point in the real world [4].

4.1.3 Virtual Reality

Kipper and Rampolla [25] describe virtual reality (VR) as a completely artificial digital environment created with computer software and hardware to construct the appearance of a real environment for a user. Furthermore, the authors describe VR as the complete immersion into a digital world based on either an actual model or an entirely artificial one. This description contrasts AR, which combines digital information with a real-world environment.

The open standard OpenXR[7] shows that the technology behind AR and VR are very similar. OpenXR tries to standardize APIs for VR and AR devices. We explain this standard in more detail in a later section.

Virtual reality is beneficial for simulation purposes, such as training new personnel. Furthermore, VR is also a good solution for practicing emergency or rare scenarios without endangering the staff or equipment. Another good use case for virtual reality is remote work with the help of robots, as can be seen in the paper of Naceri, Mazzanti, Bimbo, *et al.* [32]. Their paper discusses that remote robot teleoperations could benefit

[7] https://www.khronos.org/openxr/

from modern virtual reality interfaces. The fields of surgery, nuclear decommissioning, and disaster response environments use these remote-controlled robots.

AR, in contrast, is advantageous in onsite work. AR can blend information onto or in front of real-life elements to aid workers. Furthermore, annotating real-life objects at a work site in a 3-dimensional space also brings the benefits of more detailed and practical documentation and collaboration.

4.1.4 Mixed Reality

Another term that is less present than virtual reality and AR is called mixed reality (MR). This term has no agreed-upon definition by experts in contrast to the other two, nor does AR have a single definition that everyone agrees on according to Speicher, Hall, and Nebeling [33]. However, the authors found the following definitions:

- Mixed reality is a combination of AR and VR. Therefore these technologies are comprised of a single app or device.
- MR is a more advanced version of AR by understanding the physical environment.
- MR is a synonym for AR. Therefore it is the same.

In this thesis, mixed reality is used interchangeably with AR.

4.1.5 Limitations of AR

The systematic literature review from de Souza Cardoso, Mariano, and Zorzal [34] gives a comprehensive insight into the challenges of using AR in an industrial environment. In this review, the authors found five problem areas which we summarize in the following.

1. Hardware limitations were an issue found most often in the literature under investigation. They found that using hand-held devices is inappropriate for users in a work environment. Since both hands need to be free to work on their respective activity. Another issue is that voice commands are not always usable with the noise and safety conditions in an industrial environment.
2. Health issues were also a concern with head-mounted displays (HMD) usage. The narrow field of view of AR head-mounted displays can lead to headaches and dizziness nausea if used over a longer period. HMDs can also cause discomfort due to the device's weight on the head.
3. The authors mention that tracking methods had issues with solutions using the marker-based approach. This problem can occur because markers can not always be in a visible spot since tools or assembly components can obstruct them. Mapping the environment with markerless solutions was also linked with issues due to insufficient mature solutions. These issues were long setups and virtual object instability.

4. The authors also found insufficient projection quality and accuracy. Examples of this are virtual objects without distinct shapes and low resolutions. These problems are associated with low latency and tracking issues.

5. The development complexity of AR applications is higher than regular enterprise applications since special expertise is needed. This need for expertise leads to higher costs of implementation.

4.2 AR in Construction

This section discusses the current usage of AR in construction and where it is currently applied. Afterward, we provide a summary of recent research efforts on solutions targeting construction projects utilizing AR.

4.2.1 Introduction

As mentioned, a few useful technologies and techniques already exist that help with the digitalization of the construction industry. AR is also a technology that looks very useful in this endeavor [35]. Chen and Xue [36] mentioned that AR could help communicate designs and plans in construction processes. During operation and maintenance, AR can visualize facility information. Moreover, AR can retrieve and show valuable information directly on construction sites.

The authors of [36] found that the research on AR in construction peaked in 2013. After 2013 a decrease in interest followed, which increased again in 2018. They also witnessed that most interaction devices used for AR applications were hand-held displays. The second most used devices were head-mounted displays, and the least used approaches were projection-based or unknown devices.

Another aspect of the study from Chen and Xue [36] was their finding that most AR applications work in conjunction with BIM solutions. This integration allows users to utilize the geometric information of the building model. For example, it is possible to overlay the building model on the building or construction site. Because of this, progress monitoring can be more efficient, and defects can be detected early on. Non-geometric information can also aid workers on the construction site. For instance, rigging orders, construction schedules, and material information can be projected directly into the view of workers. Context-specific information can also increase the quality of construction work depending on the task or component.

4.2.2 AR in Different Construction Phases

Chen and Xue [36] also analyzed the usage of AR in different phases of construction projects. The authors found the least amount of papers for the design phase. These papers focused on collaborative design, design assessment and demonstration, and communication. Most AR applications target the construction phase. For this phase, the goals of the

papers included progress monitoring, defect detection, and assembly instruction. Lastly, the remaining papers belong to the operation stage. Maintenance management and operation, disaster management and emergency response, and localization and navigation were the centers of the papers for this phase.

4.2.3 State of the Art Applications

This section gives an overview of AR applications applied to construction projects. According to Chen and Xue [36] AR applications exist for the design, construction, operation, and maintenance phases. Therefore, we present a state-of-the-art AR application for each of these phases. Table 4.1 shows an overview of the discussed state-of-the-art solutions.

Multi-User Collaborative BIM-AR System

Garbett, Hartley, and Heesom [37] created a multi-user collaborative BIM-AR system for the design and operation phase of construction projects. This system fuses the information gathered from a BIM database (3D and 2D plans) with an AR application. Furthermore, it allows for addition and spatially locating information through 2D and 3D representations.

The solution developed by [37] aims to solve the issue of the increased need to communicate between the different actors of such a project. Therefore, they created their system to allow synchronous interaction with the design data through BIM processes. They emphasized that the information transfer happens bi-directional, so all stakeholders have the newest information.

The system consists of a database, an AR app, and a touchscreen table. Furthermore, the following summarizes the functionality and implementation of each component.

1. The *AR application* utilizes the Vuforia AR software development kit (SDK) for mobile devices. The authors of [37] decided to use this SDK because it supports many mobile devices and includes features such as image and 3D object tracking. Garbett, Hartley, and Heesom [37] used Vuforia in combination with Unity[8], a popular 3D engine. For the tracking approach, the authors decided to use markers. Markers were selected since they can be printed or displayed on an electronic display.

   The AR application features focus on collaboration between the project members. One such feature is viewing 3D models, which can be rotated, scaled, and moved without changing the marker. Therefore, each user has their view of the 3D model. What enables the collaboration part is the option to add annotations to points on the 3D models. The solution synchronizes these annotations in a created session between the users in real time. Mobile devices and the touchscreen table display these annotations. An icon in the 3D model represents the annotations. The users

[8] https://unity.com/

Application	**Author**	**Phase**	**Purpose**
Multi-User Collaborative BIM-AR Systems	Garbett, Hartley, and Heesom (2021)	Design and Operation	Helps with the communication between the different actors in a construction project.
Mixed Reality in a Next-Generation IoT Ecosystem	Katika, Konstantinidis, Papaioannou, et al. (2022)	Construction	OSH inspectors use this solution to aid health and safety management on construction sites.
BIM-based AR Inspection and Maintenance of Fire Safety Equipment	Chen, Lai, and Lin (2020)	Operation and Maintenance	Helps with the inspection and maintenance of fire safety equipment.

Table 4.1: Comparison of the presented state-of-the-art AR solutions.

can also assign different colors to the icon to highlight, for example, the type of comment.

2. The *BIM-AR database* consists of a MySQL database and a few PHP scripts. Annotations from users, including the color and location, are stored for their respective user sessions in the MySQL database. The PHP scripts serve as the interface to the database that exposes the retrieval and writing of information.

   User sessions are created on this BIM-AR database to allow multiple users to share views. Within a shared session, the solution synchronizes the annotations in real time.

3. Lastly, the *large touchscreen application (LTSA)* can simultaneously display different information for multiple users. For example, multiple users can display and manipulate plans and elevations. The manipulation works through the embedded touchscreen.

   The 2D information displayed on the LTSA can also be manipulated, i.e., rotated, scaled, and moved. The authors designed this interface to increase the collaboration of multiple users by enabling simultaneous working with different or the same information on a shared screen.

   A single large marker can also be displayed in this application to allow users to visualize the 3D information together through the AR application on the mobile device. However, it is also possible to show multiple markers so that users can look at the 3D models individually.

Mixed Reality in a Next-Generation IoT Ecosystem

Katika, Konstantinidis, Papaioannou, *et al.* [38] created a mixed reality solution called MR enabler with integration into a next-generation Internet of Things (NGIoT) ecosystem at a large construction site.

Samad [39] describe the Internet of Things (IoT) in a large-scale environment as a complex network of internet-connected "things". These things actuate and sense the environment and communicate this information with each other where needed. Through this information exchange, the things provide services with or without human intervention. The services are available through interfaces to make the information available anywhere and anytime. NGIoT is an initiative funded by the European Union to encourage and support the utilization of IoT for projects in Europe[9].

The system from [38] is supposed to be used by the Occupational Safety and Health (OSH) inspectors to aid health and safety management. Wearables worn by the workers gather safety information. The system then combines this information with real-time monitoring of relevant health and safety information. Timely information can be shown

[9] https://www.ngiot.eu/about/

to inspectors to help with risk identification and evaluation. Notifications and alerts inform the inspectors directly on the head-mounted device's display.

According to Katika, Konstantinidis, Papaioannou, *et al.* [38], there is an increased risk of accidents in construction work. These accidents sometimes accompany unfavorable weather conditions and long working hours without proper breaks. The authors found in their literature research that a lack of proactive and preventive measures on the construction site can lead to such incidents. These measures include the absence of risk and hazard identification, safety awareness, and proper training. Safety management is often employed to counteract these issues, improve safety processes and policies and regulate construction activities to control risks and hazards. The MR enabler tries to improve safety management operations by providing real-time information to prevent accidents actively. Furthermore, the solution tries to solve the following.

1. Detection of abnormalities that could lead to health and safety threats to construction workers.
2. Tracking of workers' locations and motion patterns. The solution then informs the management in case workers approach unauthorized or dangerous zones with the help of this information.
3. Monitoring the worker's health to prevent extreme stress, thermal discomfort, or fatigue.
4. Detection of dangerous scenarios, e.g., if a worker trips or falls, to provide a timely response.

As mentioned, the solution developed by Katika, Konstantinidis, Papaioannou, *et al.* [38] integrates with an existing NGIoT construction site. The critical interface for the AR enabler to the NGIoT system is the edge data broker. An MQTT (Message Queuing Telemetry Transport) server is the basis for this data broker. MQTT is a simple but efficient messaging protocol that works on the publish-subscribe pattern. With this server, senders can send messages to a so-called topic. Moreover, readers can subscribe to these topics to receive messages sent to this topic.

The system of the MR enabler consists of two parts. These parts are a web interface and an MR application for a head-mounted MR device. The users can use the web interface to configure the connections between the NGIoT components and the MR enabler system. Katika, Konstantinidis, Papaioannou, *et al.* [38] decided to use Microsoft's HoloLens 2 as the head-mounted device. Furthermore, the developers decided to use Unity as the development platform for this application. The MR application retrieves data from the edge data broker or long-term storage. The long-term storage contains data such as the identity, training, and assigned activities for the day for workers. MR enabler can also visualize the building models of the construction project. This MR enabler visualizes the building models by converting the building model to a format readable by Unity and

stores it on a file server. The MR solution then loads the corresponding model for the current construction site and visualizes it with the option to rotate, scale and position the 3D model. Alarms are also an essential feature of the MR enabler. These are visible inside the building model displayed through the HoloLens. The edge data broker retrieves the information for the alarms. Lastly, an interface was implemented for the HMD to generate reports of safety concerns from the OSH inspectors quickly. This interface allows the inspectors to describe the case, automatically identify the actors involved (i.e., construction workers), and take photos or videos of the incident. An example of such an incident is a construction worker not wearing proper gear to ensure safety during the operation.

BIM-based AR Inspection and Maintenance of Fire Safety Equipment

Chen, Lai, and Lin [40] created an AR solution to improve the fire safety equipment (FSE) inspection of a building. This activity belongs to the operation and maintenance phase of a building. The AR application works on tablets and aids in identifying equipment that needs checking. Additionally, the solution creates preliminary visual inspection reports to satisfy the fire safety requirements.

Fire safety equipment inspection and maintenance are essential to ensure that FSE equipment is in good working condition if needed in emergencies, i.e., to reduce fire damage. A fire safety responsible must perform such maintenance and inspections because of fire safety regulations. The authors found two types of inspections. The first type is called regular inspection, which is regulated through the Fire Services Act and specified by a maintenance plan. Non-regular inspections are the second type. This type of inspection happens when equipment fails. According to the authors, the problem with FSE inspection is that maintenance and inspection information currently has to be read from paper files. Searching and finding information on paper makes this inspection very time-consuming. The solution of Chen, Lai, and Lin [40] try to make this process faster by showing information automatically once an inspector goes inside a room under investigation. Additionally, information is blended into the camera's view if an inspector holds it toward fire safety equipment. This blended-in information not only highlights all the equipment in the room to find it easier but also enables the creation of a visual inspection report directly at the equipment.

Chen, Lai, and Lin [40] developed the AR solution as an iOS app for Apple tablets. A Firebase database was used as a backend to store 3D models and Construction Operation Building information exchange (COBie) information from a BIM solution called Revit. COBie is an information exchange specification that holds asset information, which is essential for facility managers. Stakeholders can use this standard to store maintenance information in a structured format. COBie was further extended through custom attributes to store all the information required for the fire safety equipment inspection and maintenance. The developers first convert the COBie files to Microsoft Excel files to make them human-readable before being stored in the database. Additionally, the developers export the 3D models as OBJ files, a standard open-source 3D model file

format. For the AR application to detect where the inspector is standing in the room, iBeacons[10] were used to find the tablet's location. Lastly, the inspector's approval of the preliminary visual inspection reports is sent to the database to update the information in real time for the relevant maintenance staff.

4.3 State of the Art AR Applications in Tunneling

Research regarding AR applied in the field of tunneling is very sparse. We only discovered one paper during the literature research for the specific use case of tunnel construction. The following sections summarize the paper of Zhou, Luo, and Yang [3].

4.3.1 Segment Displacement Inspection

Zhou, Luo, and Yang [3] created an AR application to inspect segment displacement during tunnel construction. Segments are elements used in mechanized tunneling supporting the tunnel. These elements are typically made out of concrete and mounted by tunnel boring machines while they are excavating. The accurate placement of these segments is critical to the structure's integrity.

Inspecting segments is a crucial part of tunnel construction. Failure to recognize displacements early on can be costly and even increase the risks of segment damage, seepage, and structural collapse. The authors mention that usually displacement detection is done with manual measurements. However, manual displacement detection is a laborious task since segments can be found at the bottom and at the top of the tunnel, which usually has a diameter of around six meters.

4.3.2 AR-based Inspection System

Zhou, Luo, and Yang [3] developed an AR-based inspection system to improve this manual inspection. This system uses a virtual baseline model from a BIM system, a 3D CAD (Computer Aided Design) drawing. The model is then superimposed in real time onto the onsite image and shown on display. Additionally, there is a control mechanism for the user to modify the location and size of the virtual model.

Zhou, Luo, and Yang [3] created a tracking system to overlap the real-world coordinates with the coordinates of the virtual model. This system works with the combination of ARToolKit[11], which is an open-source AR software library, a marker placed beside the segment edge under inspection, and a camera observing the marker and the segment. Figure 4.1 illustrates the process of this tracking system.

As the illustration shows, the superimposition of the baseline model onto the onsite video footage for inspection takes multiple steps. First, the process includes the camera calibration with the instructions given by ARToolKit. A figure in the paper of [3] shows

[10] https://developer.apple.com/ibeacon/

[11] http://www.hitl.washington.edu/artoolkit/documentation/index.html

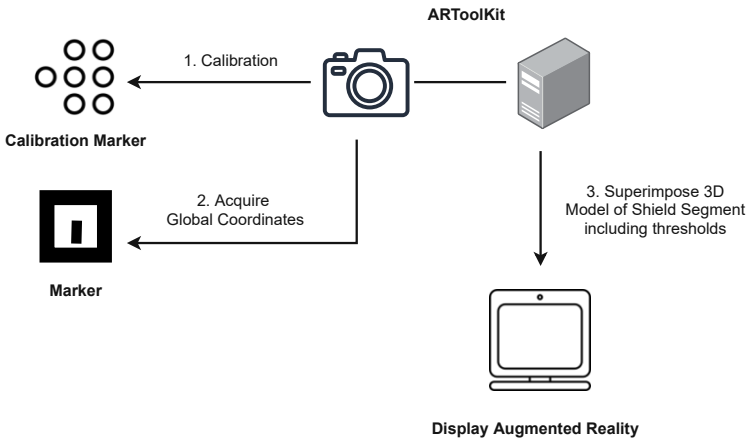

Figure 4.1: The process of the AR application created by Zhou, Luo, and Yang [3].

this calibration process. This illustration shows multiple dots on a piece of paper that has to be selected by the user. The documentation of ARToolKit includes the complete instruction for this process. Second, the operation also involves placing the marker next to the segment edge to get an exact position for the virtual model. Lastly, the model is superimposed over the collected location to show the baseline model that includes the threshold to identify displacements that do not conform to the specifications.

The following shows the proposed procedure for onsite inspections based on the AR solution (inspired by the list of [3]):

1. The procedure involves creating an inspection plan according to the standards and codes required.
2. The models for the items under inspection are stored in a BIM system and copied to the AR application. The AR application uses these models as the baseline.
3. The baseline models are linked with a marker with the help of a generator program. The next step involves printing out these markers.
4. The site worker attaches the markers to the segments under inspection.
5. The models are augmented onto the real-world image. AR glasses or mobile devices can then display these visualizations.
6. The quality inspectors can then inspect the displacement and check if it is in an acceptable state.
7. Lastly, the quality inspector reports to the site manager if the displacement is not according to code/standards. This report should then trigger a rework of the inspected segment.

4.3.3 Case Study

To test the benefits of this application, Zhou, Luo, and Yang [3] conducted a case study with an experiment. This study used the metro tunnel across the Changjiang River in China. The tunnel is a double-spool of 6 km excavated with a slurry shield machine. The segments produced by this method consist of reinforced concrete.

The authors evaluated the AR inspection solution via an onsite experiment. This evaluation included comparing the performance of the manual inspection approach and the AR solution in a user trial. These trials included four civil engineers and six researchers, and every user applied both approaches. A focus group interview was conducted after the experiment to get insights from the participants' experiences.

4.3.4 Conclusion

Zhou, Luo, and Yang [3] learned from the case study the AR solution could effectively identify unacceptable segment displacement. On average, it also took less time for the participants to inspect the segments with the AR-based solution, although it includes a setup, which the manual approach does not require.

The focus group interview revealed that most participants think the AR-based inspection system is more effective for investigating segment displacement. It was also easy for the participants to understand the AR system with a short training session. Lastly, the authors mentioned that the participants believe that a solution that does not depend on markers would be more helpful because of the inherent congested nature of tunneling project building sites.

In conclusion, the authors mention that the proposed system needs improvements for acceptance in the field. Such improvements include a better tracking system to cover more extensive ranges of the area and a reduction of markers needed. Lastly, a solution that requires no markers at all would be the best case for such an implementation, under consideration of an accurate calibration method.

CHAPTER 5

Requirements

This chapter deals with the problems and solutions proposed by the participants of the focus group interview. First, a scope is defined to limit the search for possible solutions. Thereafter, we present the actual result from the focus group interview, summarizing the problems and proposed solutions. Following that, we provide a collection of requirements collected during the semi-structured interview after the focus group interview. This part includes a more detailed summary of the problems and solutions. Furthermore, we list the refined requirements in the form of user stories. Lastly, we summarize the usability guidelines for AR applications from found literature. We present these additional requirements also in the form of user stories.

5.1 Scope

Since this study first deals with the research question of localization inside tunnels. We particularly focused on including this feature in the prototypes developed during this study. Additionally, we set limitations for collecting possible solutions found in tunneling from the focus group interview. Namely, the solutions should solve problems in the construction or operation and maintenance phase of a tunnel construction project. Additionally, the solutions should use or enhance the documentation data provided through TIMS to provide valuable information created during the construction process. The solutions should also use spatial understanding and visual indications like holograms. We chose these features to test the usability of state-of-the-art AR technologies.

Not in the scope of this study are problems solvable with simple visualizations. Examples are checklists and information displays that show general information without reference to a specific location or point of interest. We neglect such scenarios since such solutions are implementable with simple user interfaces displayed on smartphones or tablets and do not gain from the advanced features mentioned above, which AR can provide.

5.2 Idea Collection Results

The following presents the ideas collected, in combination with the problems they try to solve, collected from the focus group interview.

The first problem that was found can be assigned to the operations and maintenance phase of a tunneling project. Namely, when defects or problems are detected in a ready-built tunnel, then the operator or an expert has to investigate the incident location with information ready for that specific location. The focus group came up with the idea that through the localization inside the tunnel, the user has a button or a similar action to open the documentation for that specific round. The information should then be displayed in some window. This solution was named *maintenance inspection prototype*.

Another problem identified was that geologists must inspect the tunnel face and create a so-called geological tunnel face mapping. The captured information in this mapping is, for example, the type of rock present and cleavage. As of today, this procedure is mainly done on paper and later digitalized in specialized software. The solution proposed by the focus group was that a user could draw on a 2-dimensional transparent canvas that is placed in front of the tunnel face to make remarks and sketch information. The name chosen for this prototype is *tunnel face mapping prototype*.

A problem found regarding the construction phase was that foremen need specific information at the round they are currently working on. This information includes support definitions, sketches, survey data, and visualizations of where a round starts and ends. The focus group proposed that a solution could look similar to the maintenance inspection prototype by showing the previously defined information in the form of a window for the round at the foreman's location. This solution was named *round task information prototype*.

Another use case for maintenance work was to help contractors find the object they should be repairing or checking. Some examples of this maintenance works include maintaining drainage systems, checking the flow and CO_2 sensors, the required luminance, and fire extinguishers. A solution for this problem could be to show a path to the object that needs maintenance and then present the tasks to perform at that location. The participants also mentioned that it would be helpful for the contractors to see the live operational data that is usually only visible to the tunnel operators.

The last problem we found was regarding the work with survey instruments, so-called total stations. Usually, miners or foremen install arch support with the following procedure. For the exact placement of the support arches, workers need a total station. The workers must position the total station a certain distance from the arch. Therefore, the miners do not directly work next to each other because one has to stay near the total station. This distance makes communication sometimes hard between the two workers. The participants, therefore, proposed a solution in which one of the workers could use the HoloLens to select a position with hand gestures or directly look at it. This position is then sent to the total station, which returns the distance to the position. With this

solution, the workers could stand next to each other to make communication easier and work hands-free.

5.3 Requirements Elaboration with Experts

After the focus group interview, we selected the prototypes maintenance inspection prototype, tunnel face mapping prototype, and round task information prototype. These prototypes were selected since the interview participants emphasized their importance. Additionally, it shows the usability of different kinds of actors working in tunnel construction.

We collected the requirements for these prototypes through a semi-structured interview. During this interview, we also collected the requirements for the localization prototype inside the tunnels. The following sections include a more detailed description of the problems and the solutions found through the semi-structured interview. Furthermore, it includes a collection of requirements in the form of user stories for each prototype.

5.3.1 Localization Approach

During the interview, the participants mentioned that a markerless approach would be preferred. However, QR code markers could also be placed inside the tunnel if not possible otherwise. The interview also revealed that tunnels usually have some indicators, i.e., signs, to show the current tunnel chainage. A tunnel chainage represents the distance in meters from the start of the tunnel. For the tunnel at the ZAB, there is a tunnel chainage sign located every ten meters. Therefore, this is an excellent place to place AR markers or anchors.

Functional Requirements

Requirement L.1 As a user, I want to move freely through the tunnel and get the current location in the form of a tunnel chainage.

Requirement L.2 As a user, I want to set up the markers/anchors with the same application used to show me the information about the localization.

Requirement L.3 As a user, I want to change markers/anchors placements and even remove markers/anchors.

Non-Functional Requirements

Requirement L.4 As a user, I want to do the setup process (placing markers/anchors) only once and then make it available to other users so they can skip the setup process.

Requirement L.5 As a user, I want to place as few markers/anchors as possible to minimize the effort needed for the setup. Therefore, I only want to use a maximum of four markers/anchors per ten meters for the localization.

Requirement L.6 As a user, I want the setup to take at most 30 min per 100 m.

5.3.2 Maintenance/Acceptance Inspection Prototype

The interview revealed that maintenance inspections happen at regular intervals or after incidents have occurred. These inspections happen, for example, in tunnel highways, which [16] also mentioned. During these inspections, the facility engineers or tunnel supervisors note issues.

From the interview, we gathered that a similar process exists for the site supervision called acceptance inspection. Since this type of inspection can use the data available from TIMS, we decided to implement a prototype for this solution instead. The requirements are otherwise very similar to the maintenance inspection, and we adapted them to the insights from the interviewees. We named the prototype for this inspection *acceptance inspection prototype*.

The site supervision needs location-specific information to compare the target/actual result of the contractors' round tunneling work. This comparison of the site supervision includes checking the realized support measures and cross-sections. Since tunnels can have many support measures, this repetitive task could lead to errors when using a paper-based documentation process. During the interview, the idea of creating a report came up. This report could include a list of inspected rounds, e.g., with a button to mark a specific round as inspected to include it in the report. Additionally, notes and photos could be added to the report, including their specific positions in the tunnel, so later evaluations already have the positions and information of the proof of work.

Functional Requirements

Requirement 1.1 As an examiner, I want to retrieve all information about the round for my location.

Requirement 1.2 As an examiner, I want to create a report at the end of the inspection to capture the target/actual result.

Requirement 1.3 As an examiner, I want to include all rounds related to an acceptance inspection.

Requirement 1.4 As an examiner, I want to add notes to specific locations of the inspection site (e.g., rock bolts) so that a later evaluation of the report shows the locations of interest in the tunnel.

Requirement 1.5 As an examiner, I want to save the created report in a database.

Requirement 1.6 As an examiner, I want to photograph locations of interest and add them to the report.

Requirement 1.7 As an examiner, I want the location of interest already added to the report and grouped by the tunnel rounds.

Non-Functional Requirements

Requirement 1.7 As an examiner, I want that the report provides an overview of all the information collected, with clear separations between the text and images collected. I want the added rounds presented in a list. The notes should be in a separate list showing the tunnel chainage of the note's location, the text itself, and the photo captured.

Requirement 1.8 As an examiner, I want to be flexible to do the available actions in any order. Only the note-taking procedure should show an order in how to add notes for a specific finding.

5.3.3 Tunnel Face Mapping Prototype

The interview clarified the process of how geologists prepare a tunnel face mapping. After drilling and blasting, the workers call the geologist for inspections. The geologist first visually inspects the conditions of the new tunnel face. After that, the geologist takes a photo of the tunnel face. The geologist uses this photo as a reference for sketches of the tunnel face to indicate different types of geological information drawn and written on paper. For example, they note down the rock types, parting surfaces, and orientation of the rock and sketch the positions on the paper. The thesis of Kvasina [1] shows an example of such a tunnel face mapping sketch. Further tunneling work needs this mapping information to plan the next steps. Therefore, this process should happen as quickly as possible to avoid blocking other tunnel driving progress. This information is also essential for a comparison with the forecast of rock behavior. For example, future legal disputes with a client could require such tunnel face mappings. Such a dispute can happen in cases where rock behavior is not as expected, therefore requiring operation changes and leading to increased costs.

Functional Requirements

Requirement 2.1 As a geologist, I want to draw freely on a photo of the tunnel face to note essential findings for future construction steps.

Requirement 2.2 As a geologist, I want to be able to draw lines on the photo of the tunnel face to note findings that are important for future construction steps.

Requirement 2.3 As a geologist, I want to add text and annotate the drawings created on the photo of the tunnel face.

Requirement 2.4 As a contractor, I want to save the created annotation with the tunnel face visible in the background for the documentation process.

Requirement 2.5 As a contractor, I want the location of the tunnel face documentation saved as additional information for the mapping. The location is helpful to avoid mix-ups when I search for a specific tunnel face mapping. The tunnel face mapping should be findable with the location.

Requirement 2.6 As a geologist, I want to edit the annotations and drawings on a computer after the mapping.

Non-Functional Requirements

Requirement 2.7 As a geologist, I want to be able to start annotating a tunnel face in less than 5 min. That means the setup should take at most 5 min.

Requirement 2.8 As a geologist, I want the menu of annotation tools to be on one page and to change the tools with less than three clicks.

5.3.4 Round Task Information Prototype

During the semi-structured interview for the prototypes, we found that AR does not provide any benefits for the round task information prototype. It is not practical because localization is not helpful since the foremen work at the newest round. Therefore, it is always evident in which round the users currently work. An application that sorts the rounds according to the planned work is more straightforward than an AR solution. Such an application could give the user the information for the current tunnel round. Smartphones or tablets could display this information in a digital form. Another idea was to give the foremen an overview of the tasks they have to carry out with embedded information about the current round. This solution could provide a hands-free set of instructions to aid workers in what to do and how to do it. However, the interview revealed that the tasks of foremen do not always have a predefined order. For example, if problems arise, they might need to be resolved first, i.e., the order of the tasks can change according to the priority. Therefore, a strict set of tasks visualized through AR is probably not helpful for this work.

5.4 Usability AR Guidelines

Additionally, we consider input from literature concerned with the usability of AR. We decided to search for this input to further increase the developed prototypes' usability.

The following presents a summary of the found guidelines for AR solutions. Furthermore, we include additional non-functional requirements in the form of user stories that should be useful for developing the prototypes.

One important aspect of AR solutions is dealing with effects such as simulator sickness observed by Kaufmann and Dünser [41]. The authors found possible reasons for this effect. Namely, these reasons include low frame rate, lag, and poor fitting of the head-mounted devices. Therefore, special attention should be on maximizing the frame rate. By showing less computationally intensive information, the application should avoid lag as much as possible. Furthermore, the head-mounted devices should be adjusted to the user to ensure they fit comfortably. The implementation of Knopp, Klimant, and Allmacher [42] shows another rough guideline. They mentioned a strong focus on making the AR application as hands-free as possible. They decided on this design so the workers could work hands-free. Another design choice was to keep the limited field of view of the HoloLens manageable. Therefore, they created multiple menus that the user can click through. Lastly, the authors mention that a user-centric interaction design would be the best approach to increase the usability of AR solutions. This approach is similar to the design science research of Hevner, March, Park, *et al.* [7]. Therefore, remarks from users in the usability tests will be made available in this thesis for future research or iterations of the proposed solutions.

Requirement D.1 As a user, I want a steady frame rate of at least 15 frames per second to decrease the chance of simulator sickness. Hecht [43] identified this number, who looked at previous VR implementations and found that 4–12 frames per second make users feel sick.

Requirement D.2 As a user, I want windows only to include information relevant to a specific topic to keep my field of view manageable.

Requirement D.3 As a user, I want the information to be displayed close to me, with the fewest interaction steps necessary to free my hands for other tasks. Therefore, I want windows to float automatically in the field of my view where appropriate.

CHAPTER 6

Tools & Approaches Applicable to the Prototypes

This chapter deals with the tools, frameworks, and strategies for developing AR applications. That includes the devices displaying AR and the available development platforms and techniques. Lastly, this chapter will also discuss the chosen technologies for the high-fidelity prototypes developed during this thesis.

6.1 Localisation Techniques

Localization techniques are essential in this thesis since all solutions rely on them. In AR, there are three promising options for finding the user's position in the real world. These approaches are *marker-based*, *spatial anchors-based*, and *GPS-based*, and we will explain them in the following.

6.1.1 GPS-based

GPS (Global Positioning System) is a system of satellites. This system transmits radio signals to the ground, which devices can passively receive to calculate the current position and velocity of the device as well as the time, as explained by Abulude, Akinnusotu, and Adeyemi [44]. AR applications can use this technology to overlay different kinds of information or visualizations at specific positions in the world. Such solutions are often accompanied by other sensors, such as inertial sensors, to increase localization precision. The research of Unal, Bostanci, Sertalp, *et al.* [45] and Delić, Domančić, Vujević, *et al.* [46] shows such an approach.

The advantages of GPS are that it is cheap to develop and use. Devices only need little computational power to work with GPS sensors. The disadvantages are that the

accuracy is around a few meters depending on the environment[1]. Also, GPS localization is impossible in tunnels since the signal is attenuated by rock or concrete.

6.1.2 Marker-based

Another common approach to detecting positions in the real world is using markers that a camera can detect. Typical examples are QR codes or images. Such solutions can then virtually overlay these markers with visualizations and information. For example, the works of Gherghina, Olteanu, and Tapus [47] show an early adaption of using QR codes in AR. Gherghina, Olteanu, and Tapus [47] used QR codes because they can be used to detect surfaces by identifying the corners. Additionally, AR applications can read information stored on such codes to look up data from a server.

The paper from Boonbrahm, Boonbrahm, and Kaewrat [48] shows another example of marker-based AR. In this paper, the authors use markers to measure the size of a room and show a high precision with a 3 % error rate compared to manual measurements.

The advantage of markers for localization is that they can already store positional information about themselves, so precise placement can make the localization setup very effective. These markers can also store other information about the given positions, like specific visualizations or information from a server.

A disadvantage of this technique is that users have to place the markers precisely in the target environment. Depending on the size, the users have to place many markers. This marker placement is a time-intensive expenditure since the positions must be measured, and then all markers must be placed. Lastly, markers sometimes are not allowed to be put up, or it is not easy to attach markers to the given environment.

6.1.3 Markerless Anchors

Markerless AR is a method that, as the name suggests, works without any visual markers and does not need additional devices for localization. Such a solution utilizes so-called anchors to achieve this without needing physical markers. Most popular AR frameworks such as ARCore, ARKit, and MRTK[2] (Microsoft Mixed Reality Toolkit) make use of this approach. Anchors are similar to markers, but the markers are created virtually by capturing features from the real world. According to the Microsoft Research Blog article from Delmerico, Oleynikova, Nieto, *et al.* [49], SLAM is the approach behind these anchors.

The research of Varelas, Pentefoundas, Georgiadis, *et al.* [50] shows some examples of this approach. The authors use markerless anchors for an indoor positioning system to navigate users through an unknown environment. The solution developed by Chacko and Kapila [51] is another example. They developed a human-robot interaction interface

[1] https://www.gps.gov/systems/gps/performance/accuracy/

[2] https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/architecture/overview?view=mrtkunity-2021-05

with AR. The markerless approach was applied to this solution to interact with this robot and set anchors to define essential points. The authors combined this technique with a marker approach for defining the robot's pose and the working space. Chacko and Kapila [51] applied the markerless approach, in addition, to reducing the physical markers needed to a minimum.

The advantages of this method are the lack of placed markers needed for the localization. This lack of markers means that no additional material is needed, and only the AR device suffices for the setup and regular usage of the application. The disadvantages are that anchors are not always as precise as physical markers. This precision heavily depends on the environment since feature-rich environments make it easier to find positions again. At the same time, plane surfaces can be harder to detect and differentiate from each other. Unfortunately, we could not find any research on localization utilizing AR in tunnels. This thesis wants to provide answers for this use case to clarify if the detection works well in such an environment. The tunnel lining is usually not very feature-rich and has a large surface.

6.2 Augmented Reality Devices

This section summarizes different AR devices' specifications, viability, and usage areas. The focus lies on currently available devices suitable for this thesis's proposed solutions.

6.2.1 HoloLens 2

HoloLens 2[3] is the second generation of Microsoft's Mixed Reality HMDs (Head Mounted Devices). This MR device can project three-dimensional objects into the wearer's field of view, extending reality with helpful information or depictions. Furthermore, the HMD also detects hand movement, can track eye movement, and understands natural language, which makes it easy to interact with projected information. HoloLens 2 consists of a standalone computational unit that runs the *Windows Holographic OS*, a customized version of Windows 10, and the following sensors according to the hardware specifications from Microsoft[4]:

- Four visible light cameras for head tracking
- Two infrared (IR) cameras for eye tracking
- 1-MP (megapixel) time-of-flight depth sensor for depth mapping
- Accelerometer, gyroscope, and magnetometer for inertial measurement
- Camera with the capability of 8-MP stills and 1080p30 for video

[3] https://docs.microsoft.com/en-us/hololens/hololens2-hardware
[4] https://docs.microsoft.com/en-us/hololens/hololens2-hardware

Regarding accuracy, we found the following two papers.

Indoor Mapping Eyewear: Geometric Evaluation of Spatial Mapping Capability of HoloLens, by Khoshelham, Tran, and Acharya [52], goes through the process of studying the spatial mapping capability of the HoloLens in large indoor environments. The authors achieve this by comparing the results with terrestrial laser scanners, which can achieve millimeter-level accuracy. Albeit this was not using the second generation of the HoloLens, the HoloLens 2 probably performs similarly or better than the previous version since it also uses a time-of-flight depth sensor. However, Microsoft does not give more detailed information about this sensor for the first-generation HMD.

The comparison results in *Indoor Mapping Eyewear: Geometric Evaluation of Spatial Mapping Capability of HoloLens* is a 3D mesh. Khoshelham, Tran, and Acharya [52] focus on three geometric aspects to evaluate the captured data. These aspects are the following:

1. *Coverage* shows how much of the surfaces of the reference 3D model are covered by points in a point cloud/mesh of the HoloLens.

2. *Global Correctness* depicts how the global shape of the captured environment is consistent with the real-world dimensions and layout of the environment. The authors evaluate this by comparing the captured 3D mesh with the data of a laser scanner and a reference 3D model and computing the offset distances.

3. *Local Precision* indicates the precision of single 3D points collected by the HoloLens. Khoshelham, Tran, and Acharya [52] compute this value by fitting planes to points captured on planar surfaces. The root-mean-squared error of plane fitting is then the result.

The evaluation of Khoshelham, Tran, and Acharya [52] shows that the local precision was 2.25 cm. The authors measured this precision by the root-mean-squared error of the plane fitting over 4002 points on ten planes. The global correctness was 5.68 cm, achieved by comparing the root mean squared distance between 3409 mesh vertices of the HoloLens and ten laser-scanned planes that corresponded to the same area. Lastly, the coverage was rated as very high compared to the laser scan, especially since the HoloLens only needed a few minutes compared to the multiple hours of the laser scanner.

Soares, B. Sousa, Petry, *et al.* [53] worked on a similar study of the accuracy of the second generation of the HoloLens in *Accuracy and Repeatability Tests on HoloLens 2 and HTC Vive*. In this study, the authors compared the accuracy of hand tracking of the HoloLens 2 to device tracking of the HTC Vive[5] and as a ground truth, a motion capture system called OptiTrack[6], that has a sub-millimeter accuracy. The difference between these devices is that the HTC Vive is a virtual reality headset that uses a controller to

[5] https://www.vive.com/us/
[6] https://optitrack.com/

track hand positions. In contrast, the HoloLens can track hands directly without any additional device. OptiTrack is a motion capture system that uses multiple infrared cameras to track objects in a specific area. Customers use this technology for applications such as movement science, tracking of drones, and motion capturing for video games and movies.

Soares, B. Sousa, Petry, *et al.* [53] tested the accuracy of the tracking capability by various means, e.g., when the tracking object was stationary, moving at different velocities, and when the HoloLens 2 was tracking the hand outside the center of the field of view. The authors calculated the comparison results by looking at the delay and accuracy of the position of both devices. This calculation resulted from synchronizing the timestamps of the captured data with the ground truth data and comparing the results.

From these tests Soares, B. Sousa, Petry, *et al.* [53] found out that the hand tracking of the HoloLens 2 performs significantly worse than the HTC Vive. The authors expected this result since the HTC Vive uses two external infrared cameras to acquire the tracked object's position. However, they also found that the tracking performance works best if the hand is inside the vision field. Also, the hand tracking was rated as acceptable by the authors for tasks such as gesture recognition and painting applications, so in areas where lower accuracy is enough.

The HoloLens 2 is very interesting because it is one of the only wearable augmented/mixed reality devices with features such as projecting three-dimensional objects in a way that its depth is understandable in the real world. Other features that make this device viable are scanning environments to distribute their 3D model and using this information for users to place information into not yet scanned environments. Especially since usually more expensive and laborious scans are needed to achieve similar results, such as laser scans. For comparison, a HoloLens 2 costs € 3,849[7] at the time of writing, while laser scanners usually start at over 10 thousand Euro, such as the *Faro Focus Laser Scanner*[8] and *Leica BLK360*[9].

A drawback of this MR device is that the battery life is only around 2-3 hours of active usage, so depending on how long tasks have to be, more than this might be needed, although an additional power bank could help. Another point is that the field of view of the HoloLens 2 is 52°, which is an improvement to the 30° of the first generation, but it still sometimes makes it hard to get the whole picture of projected information. However, focusing on the user experience while developing AR applications can mitigate this issue.

[7] https://www.microsoft.com/de-de/hololens/buy
[8] https://www.faro.com/de-DE/Products/Hardware/Focus-Laser-Scanners
[9] https://leica-geosystems.com/en-gb/products/laser-scanners/scanners/blk360

6.2.2 Magic Leap

Magic Leap 1[10] is very similar to the HoloLens 2. It is also a wearable device capable of projecting three-dimensional models into the field of vision of the user. The AR device can also recognize speech commands and can utilize hand tracking. However, the website[11] of Magic Leap demonstrates that their focus in hand tracking lies on gesture recognition, instead of using it to manipulate 3D models in AR. This focus on gesture recognition does not mean that developers cannot implement object manipulation with the given information from the hand-tracking API. However, it is less integrated than in the HoloLens 2. Magic Leap's primary mechanisms to manipulate and work with 3D objects in AR are the six degrees of freedom (6-DOF) controller and the smartphone app, which can act as a remote.

Another distinction from the HoloLens 2 is that the computational unit is not located in the head-mounted device. Instead, an externally connected unit, which Magic Leap calls *Lightpack*, holds the computational unit. This Lightpack is attached with a clip, for example, onto the pockets of the user's pants. The computational unit runs Lumin OS, deriving from Linux and the AOSP (Android Open Source Project). Magic Leap's website[12] lists the following included sensors in this device.

1. Accelerometer and magnetometer for inertial measurement
2. Camera with the capability of: 1080p30 for video

Unfortunately, Magic Leap's website only provides little information about the sensors used for depth measurements. TechRepublic[13] mentions that the Magic Leap contains eye-tracking infrared sensors and an infrared dot projector for room measurements. However, this article lists no sources.

Magic Leap also does not provide much information concerning the accuracy of the Magic Leap 1 regarding the various tracking capabilities. One paper from Schneider, Biener, Otte, *et al.* [54] evaluated the hand tracking accuracy from various AR and VR devices, including the Magic Leap. The result showed that the hand-tracking accuracy of the HoloLens is significantly more accurate than the Magic Leap.

The price of around € 2,120 makes the Magic Leap a cheaper alternative to the HoloLens 2, which also has very similar specifications. The field of view is also in a similar range of 50°. The usage time of this AR HMD is rated at 3.5 h of continuous use, so longer than the HoloLens 2. An advantage of this device is that its weight is more distributed because of the external computational unit, which reduces the weight on the head of the wearer.

[10] https://www.magicleap.com/magic-leap-1
[11] https://developer.magicleap.com/en-us/learn/guides/lumin-sdk-handtracking
[12] https://www.magicleap.com/magic-leap-1
[13] https://www.techrepublic.com/article/magic-leap-1-augmented-reality-headset-a-cheat-she

The disadvantage of the Magic Leap is the lack of research utilizing this device. Therefore, more research is needed to compare the device's capabilities in real scenarios. Also, the device focuses on using a controller to interact with the virtual world. Holding a separate device can make it difficult for specific scenarios where the user cannot carry an additional controller. While in contrast, short interactions via hand tracking would be possible.

6.2.3 Mobile Devices

Smartphones today are capable of visualizing AR. Both Android and iOS devices are capable of utilizing AR. These devices use regular cameras to visualize virtual 3D models with the help of markers or algorithms. Smartphones with additional sensors also help with the localization process and mapping of the environment.

One example is the iPhone/iPad Pro since 2020, including a LiDAR (Light Detection and Ranging) sensor to detect depths. Tavani, Billi, Corradetti, *et al.* [55] evaluated the iPhone 12 Pro regarding various aspects that could help geoscience. They also looked at the LiDar sensor since it can improve camera focusing and enable 3D outcropping. The authors found that the accuracy of the iPhone for distance measuring is in the range of 1 cm, which Apple also claims. They also found that the maximum range in which objects or environments are scannable is around 3 m. The claimed maximum range under perfect conditions is 5 m. For the 3D outcropping, the authors found a scaling error of approximately 5 % present for the models they scanned.

Other examples are Samsung's Galaxy S21, Galaxy Note 20, and LG's G8, which contain a time-of-flight sensor. Unfortunately, it is not certain that Samsung will add this sensor into future models since the Galaxy S22 will no longer contain this sensor[14].

Scargill, Premsankar, Chen, *et al.* [56] created a qualitative comparison of virtual object stability with markerless AR in various smartphones and the HoloLens 2 as a baseline. With virtual object stability, the authors mean how far objects drift from their placement position in the real world. This drift happens when the camera is directed at a virtual object in a placed location after the object was shortly not in the camera's focus. The issue with this drift is that it can lead to an offset of the object's original position to its placed position. The authors compared four different smartphones, including an iPhone with LiDar, one without LiDar, and a Samsung Galaxy with a ToF sensor. This study revealed that specialized cameras in smartphones improve spatial stability. However, the spatial robustness of 3D objects positioned in markerless AR environments is still a problem in smartphones.

The prices of the phones range from android smartphones that do not include special sensors starting from around € 180 for a Samsung Galaxy A5 (2017) to Apple's Ipad Pro, which costs at least € 1.049. The supported devices page[15] shows the devices compatible with the AR framework from Google called ARCore.

[14] https://english.etnews.com/20210419200003
[15] https://developers.google.com/ar/devices

The significant advantage of using smartphones for AR is the wide availability and that they are generally cheaper than dedicated devices. This availability also makes it easier to deploy AR applications if users own a compatible smartphone. Lastly, smartphones with specialized sensors can enhance AR spatial stability while still being cheaper than dedicated AR headsets.

The disadvantage of smartphones as AR devices is that, depending on the application, spatial accuracy might not be high or robust enough. Depending on the environment, holding a smartphone while carrying out additional activities can be cumbersome using an AR application, e.g., if the information has to be written down or repairing activities must be carried out while getting spatial information about the worked-on object.

6.3 3D Development Platforms

This section discusses the available development platforms used to develop AR applications. The first part also includes a description of OpenXR, an API used by both discussed development platforms. Lastly, this section summarizes both Unity and the Unreal Engine.

6.3.1 OpenXR

OpenXR is an open and royalty-free standard that tries to unify access to XR platforms and devices. This standard was created by the Khronos OpenXR Working Group, a consortium supported by multiple large XR companies such as Microsoft and Meta, according to the press release of the Khronos group[16].

The problem OpenXR tries to solve is that vendor-specific APIs, which expose standard functionality, are often very different from each other. This divergence makes it harder for developers who target multiple platforms or hardware because they have to account for every API difference. OpenXR solves this issue by creating a unified application programming interface that exposes the functionality of each vendor-specific runtime system. A standard set of objects and functions included in this API are tracking, frame timing, input, and application lifecycle.

6.3.2 Unity

Unity[17] is a real-time 3D development platform for different applications. Examples of applications developed in Unity exist in gaming, automotive, architecture, and construction. For AR, multiple frameworks are available to develop applications, most of which are compatible with Unity. This generic platform makes it easy to develop different applications and AR applications and publish them to different types of devices. Unity

[16] https://www.khronos.org/news/press/khronos-releases-openxr-0.90-provisional-specification-for-high-performance-access-ar-vr-platforms-and-devices

[17] https://unity.com

supports deploying to Android/iOS devices, most AR/VR devices, browsers, and other less important devices for AR.

The primary programming language of Unity is C# which is used in so-called scripts to control the different aspects of a 3D real-time application. If a developer creates scripts[18] in Unity, then the editor creates a new file with a C# class deriving from the built-in class called `MonoBehaviour`. `MonoBehaviour` is the connection with the Unity engine that allows interaction with other scripts and components in an application. Additionally, to `MonoBehaviour` scripts, it is possible to implement standard classes and use them in scripts like in a typical C# project class.

Unity uses scenes in which content, such as 3D objects and scripts, are embedded to form an application. So everything from a user interface to code that interacts with 3D models lives in such a scene. An object in a scene is called a `GameObject`. A `GameObject` is the base class of all entities displayed in a scene. This class already contains properties such as its transformation, name, and tag. The Transformation class is an important aspect that holds an object's position, rotation, and scale. Another essential feature of `GameObjects` is that they use parent-child hierarchies to group `GameObjects` together. This hierarchy property means that a `GameObject` can hold other `GameObjects`. Scripts and other properties, such as textures, are components directly attached to `GameObjects`. Components are not `GameObjects` themselves. A scene, therefore, comprises `GameObjects` holding other `GameObjects` with components attached to them. An example of an object in a Unity scene is a `GameObject` with a `MeshRenderer` component (which renders meshes) and a `TextMesh` component. This `GameObject` can display text to the user if the camera is pointing at it.

6.3.3 Unreal Engine

Unreal Engine[19] (UE) is very similar to Unity. This engine is also a real-time 3D development platform to create applications in similar areas as Unity. For example, the gaming, architecture, and automotive industries also use it. Lastly, the supported platforms for Unreal Engine also include iOS/Android and VR/AR devices.

In UE, there are two options for creating logic[20] in an application. First, there is Blueprint visual scripting, which, as the name suggests, uses visual scripting by creating nodes that define inputs and outputs to which other nodes can connect. The other option is programming in C++ by creating classes inherited from `UObject`. This approach is similar to Unity's one. Developers can mix the two options using C++ classes in blueprints and vice versa. C++ has better structure and performance, while blueprints are suitable for prototyping and implementing generic cases built in C++.

[18] `https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html`
[19] `https://www.unrealengine.com`
[20] `https://docs.unrealengine.com/4.27/en-US/Basics/UnrealEngineForUnityDevs/`

UE works very similarly to Unity regarding how a scene is structured[21]. In UE, a scene is called a map file. `Actors` are the same as `GameObjects` in UE. Lastly, `Actors` can also hold components.

6.4 AR Frameworks

This section explores available frameworks to implement AR applications. First, there is a summary of some of the most popular AR frameworks. Finally, this section includes a comparison table highlighting the mentioned frameworks' differences.

6.4.1 MRTK

MRTK (Mixed Reality Toolkit)[22] is a combination of tools and an application framework developed by Microsoft. Developers can use it to build VR and AR applications for platforms such as OpenXR, OpenVR, and WMR (Windows Mixed Reality). This wide range of platforms makes it possible to deploy AR applications to multiple devices such as the HoloLens (version one and two), iOS and Android devices, and the magic leap with the MRTK-MagicLeap extension[23]. Lastly, the MRTK can be used in Unity and UE development platforms.

MRTK offers various functions and tools for the development of AR applications. Some examples worth mentioning include the following.

1. Access to the spatial awareness system to get information and functions for scanned environments. This access means that 3D model information can be extracted and used for different calculations and visualizations.

2. It also offers a diagnostic system that can display useful performance information about the device.

3. Different UX (user experience) building blocks are available to create interactable menus or show different information, e.g., buttons, dialogs, and sliders.

6.4.2 ARKit

ARKit[24] is an AR framework developed by Apple to create applications for iOS devices. Devices supported by this framework include iPhones since version six and all iPad Pros. UE and Unity can utilize ARKit with the ARKit XR plugin[25].

[21] https://docs.unrealengine.com/4.27/en-US/Basics/UnrealEngineForUnityDevs/

[22] https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/architecture/overview?view=mrtkunity-2021-05

[23] https://developer.magicleap.com/en-us/learn/guides/unity-mrtk-project-setup

[24] https://developer.apple.com/documentation/arkit/#overview

[25] https://docs.unity3d.com/Packages/com.unity.xr.arkit@4.1/manual/index.html

Some of the special features that are available through this framework are the following:

1. Motion Capture makes it possible to use the information about the movement of bones and joints of a person.
2. Scene Geometry provides information about the scanned environment, such as floors and furniture. This information can be used to interact with these objects and even export their meshes as files.
3. The depth API, combined with the Scene Geometry capability, increases the precision by utilizing the LiDAR sensors in the devices that ARKit supports.

6.4.3 ARCore

ARCore[26] is an AR framework created by Google. This framework is targeted primarily at smartphones, including devices from iOS and Android. ARCore supports more iOS devices than ARKit because it also supports non-Pro iPads. ARCore is available for multiple development platforms, including Unity, UE, and Android Studio.

The following shows some useful features included in this framework.

1. Cloud Anchors allow sharing information persisted at a location in the real world to be shared with other devices, so they see the same information at the exact location.
2. Hit-Test allows casting a virtual ray from the phone to the real world and can return the intersecting real-world or virtual objects. This feature makes it possible to interact with this information.
3. The Augmented Images API allows detecting images such as posters or product packaging to augment them with additional information or effects.

6.4.4 Vuforia

Vuforia Engine is an AR library developed by PTC Inc. It supports iOS and Android smartphones, Microsoft Surface devices, and eyewear devices such as HoloLens (1 and 2), Magic Leap, and RealWear. An exciting aspect of Vuforia is that it can be used in other frameworks to enhance and add additional features. An example of this is the usage with MRTK[27] to add the feature of recognizing physical objects, such as vehicles and machines, by providing CAD and other 3D models. Regarding platform support, there is currently only official support for Unity.

The following contains a list of some useful features of Vuforia.

[26] https://developers.google.com/ar/develop

[27] https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/vuforia-development-overview

1. Cylinder Targets: Vuforia can recognize image targets shaped around cylindrical objects.

2. Multi Targets: allows using multiple image targets to detect objects from various sides (e.g., objects in the form of a box) and help with corner visualizations. For example, they can show different information on each side of a cereal box.

3. Model Targets: a valuable feature for recognizing physical objects with preexisting 3D models.

Vuforia is the only framework presented in this section with a licensing fee for the more advanced features.

6.4.5 Comparison

ARKit and ARCore support the least amount of devices but focus heavily on portable devices (i.e., Smartphones, Tablets). Vuforia has the most significant number of officially supported devices but lacks markerless and surface detection by relying on third-party frameworks or specific hardware, which restricts the supported devices again. Mixed Reality Toolkit has similar restrictions to Vuforia for the surface detection and markerless approach but supports everything for the HoloLens 1 & 2.

Table 6.1 shows a general comparison overview of the different AR frameworks presented in this section. This table helps identify which AR frameworks are most suitable depending on the targeted devices and localization technology. All presented AR technologies can detect markers in some form.

6.5 Tools & Approaches for the Prototypes

The following will explain the tools and approaches used for the prototypes built for this thesis.

The solutions need adequate accuracy to be helpful in tunnel construction. Therefore, we decided to rely on marker-based and markerless localization. As already mentioned, GPS-based localization is impossible in tunnels because the signal cannot reach there. While developing the prototypes, it is essential to test the applicability of markerless localization and substitute it with the marker-based approach, if necessary. Since the AR applications developed should support different stages of the lifecycle of a tunnel, it may be necessary to keep anchors while the environment around it is changing. This possible change makes it harder for devices to recognize markerless anchors since the detection relies on visual features from the environment.

The device we selected for the prototypes is the HoloLens 2. We selected an explicit AR device because of the superior localization technology compared to mobile devices. Additionally, such devices include specific sensors and hardware to improve the AR

Framework	**Pricing**	**Device Support**	**Markerless**	**Scene/Surface Detection**
MRTK	Free	Smartphones (Android, iOS), AR Devices (HoloLens), VR Devices (Oculus, Leap Motion)	Yes (HoloLens 1 & 2, or ARKit/Core)	Yes (HoloLens 1 & 2)
ARKit	Annual Developer Program account (US$99) for distribution	Smartphones/Tablets (iOS)	Yes	Yes
ARCore	Free	Smartphones/Tablets (Android, iOS)	Yes	Yes (limited capability)
Vuforia	Free Version and a Pro Version but the price has to be discussed	Smartphones/Tablets (Android, iOS, Microsoft Surface), AR Devices (HoloLens 1 and 2), Magic Leap, RealWear, Vuzix)	Yes (but needs ARKit/Core or other 3D scanning devices for areas)	Same as for markerless

Table 6.1: Comparison of the different AR frameworks.

capabilities/features. A head-mounted device also has the advantage of having the hands free. Furthermore, looking around without holding a screen in an uncomfortable position while viewing information is more accessible.

Microsoft's HoloLens 2 was chosen over the Magic Leap because more research is utilizing the HoloLens, making it a more reliable option. Also, research exists on the AR device from Microsoft, presenting the accuracy of the tracking capabilities and even comparisons to other VR/AR devices. Such a study does not exist for the Magic Leap.

There is no significant difference between the two development platforms, Unreal and Unity, for developing AR applications. Both platforms support most AR frameworks. Hence, Unity was chosen as one of the two platforms with no particular advantage over the other.

Lastly, we chose the MRTK from Microsoft as an AR framework for the prototypes. Since Microsoft also develops the HoloLens, MRTK is the most tailored framework for this device. Vuforia would be another candidate, but it lacks features that are only accessible with the pro version and does not support some features, such as plane detection on the HoloLens. MRTK can utilize Vuforia, but the additional features provide no benefits for the prototyped solutions. MRTK provides a lot of ready-to-use components to build user interfaces and enable different interactions, such as hand gestures. It also provides localization solutions, such as for local images or areas in specific environments. However, it also contains the so-called World Locking Tools[28] to provide a localization solution that can also handle larger environments. Finding an optimal solution from the provided MRTK localization approaches is an integral part of the prototypes and will be discussed later.

[28] https://docs.microsoft.com/en-us/mixed-reality/world-locking-tools/

CHAPTER 7

Implementation

This chapter explains how we implemented the prototypes for this thesis. First, we explain the localization prototype and what type of localization from the MRTK framework we used. After that, we explain the acceptance inspection prototype. Moreover, this section shows the implementation approach for the tunnel face mapping prototype. An overview of all the functions mentioned in this chapter can be seen in Table 1.

7.1 Localization Inside a Tunnel

We found a few approaches to implement the localization solution inside the tunnel with the HoloLens. Namely, these approaches are spatial anchors, markers in the form of QR codes, and the World-Locking Tools framework. However, spatial anchors and AR code markers have the disadvantage that only within a range of three meters around them a user can achieve a high-precision localization[1,2].

World-locking Tools (WLT) is a framework created by Microsoft that utilizes the automatic placing of spatial anchors to lock the coordinate system from Unity to the physical world[3]. This locking of coordinates means that holograms placed in a location in the real world can later be loaded again at the same position. As already mentioned, however, spatial anchors need unique features to be detected again by cameras to locate themselves. A tunnel is usually a feature-poor environment, which could lead to wormholes where the HoloLens thinks it is in another similar location and therefore shows the wrong position in Unity's coordinate system[4]. Therefore, we kept this problem in mind during the tests

[1] https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-anchors

[2] https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/qr-code-tracking-overview

[3] https://microsoft.github.io/MixedReality-WorldLockingTools-Unity/README.html

[4] https://learn.microsoft.com/en-us/hololens/hololens-environment-considerations

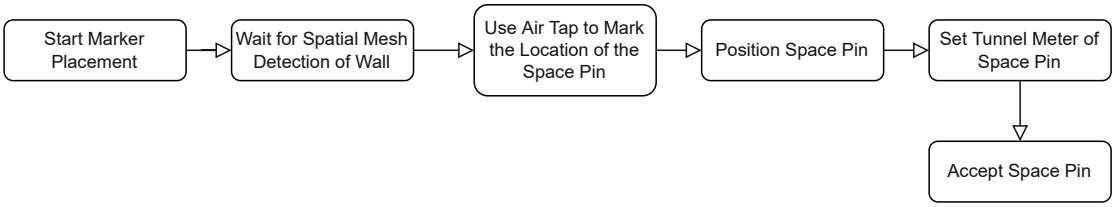

Figure 7.1: The process of placing space pins in the tunnel with the prototype.

in the tunnel.

We decided to use a feature from the WLT called space pins[5] to mitigate the wormholes problem and increase the localization precision. Space pins solve the scaling error that occurs when traveling a more considerable distance with the HoloLens, which introduces positioning errors of ±10 %. That means, if a user walks 10 m with the HoloLens, the distance traveled in Unity's coordinate system can be 9 m–11 m instead. Space pins are also spatial anchors but have a unique purpose within the WLT. According to this point, these anchors are used to realign the coordinate system. WLT achieves this by interpolating the coordinate system from Unity and the space pin. This interpolation reduces the scaling error if an environment contains enough space pins. Microsoft mentions that a spacing of 10 m between space pins in an office environment can reduce the accumulated error from 10 cm–20 cm to just millimeters between those spaces.

With the information on how the WLT and space pins work, we decided to place the space pins every 10 m inside the tunnel. At the ZAB tunnel, there are convenient tunnel chainage signs located at every tenth meter. The tunnel chainage is the distance along the tunnel axis in meters from the start of a tunnel. Figure 7.1 shows this process. First, the user has to start the marker placement by pressing a button. Following that, the user has to wait for the wall detection of the spatial awareness system[6] created by Microsoft. The spatial awareness system can create 3D meshes to represent the geometry of the real-world environment. This detection is necessary for the next step. Namely, a space pin marker can be placed directly at the intersection of the wall and the air tap. An air tap is a hand gesture of tapping the index finger and the thumb together. The user has to hover their hand in front of the HoloLens to aim with this air tap. This action creates a beam that starts at the hand of the user and ends at the nearest 3D object detected in a straight line. Figure 7.2 shows an example of this beam. After the user positions a space pin at the air tap location, they can still move it to fully cover the tunnel chainage sign. Once the space pin position is correct, setting the real-world tunnel chainage for this space pin is possible. Lastly, the user has to accept the space pin to finish the creation.

[5] https://learn.microsoft.com/en-us/mixed-reality/world-locking-tools/documentation/concepts/advanced/spacepins

[6] https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/spatial-awareness/spatial-awareness-getting-started?view=mrtkunity-2022-05

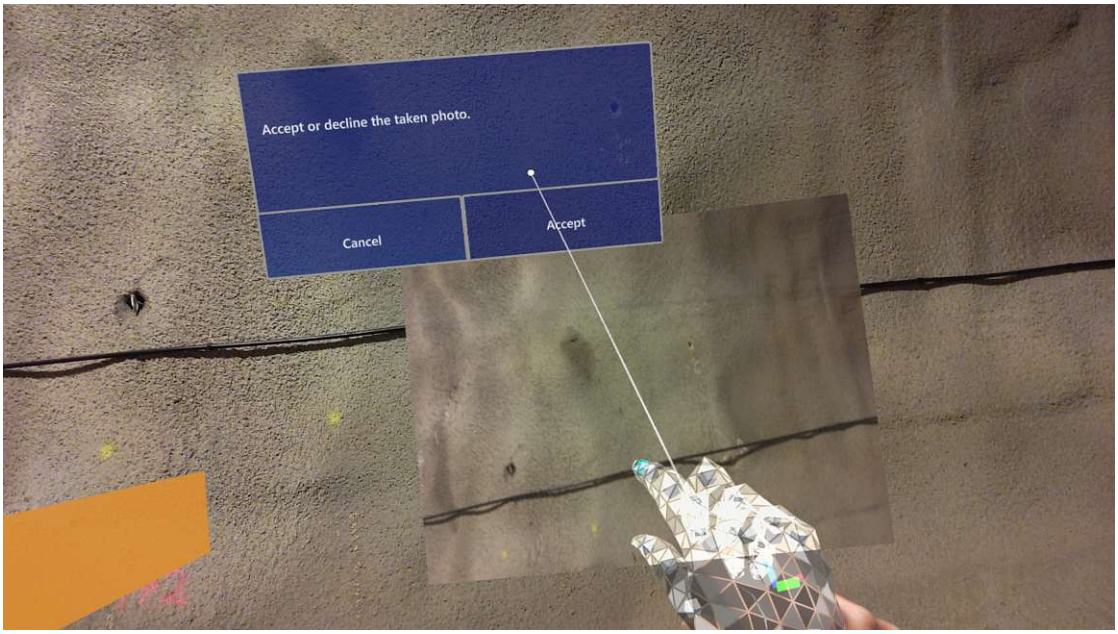

Figure 7.2: An example of the HoloLens creating a beam from the hand of the user, used for air taps.

We decided to use space pins as spatial anchors and QR code markers for the prototypes. Both types of markers ensure that the solution can realign the coordinate system in case the spatial anchors are not detectable in tunnels since QR codes can always be detected as long as they are at a certain distance.

Figure 7.3 illustrates the three-tier architecture of the localization prototype. Furthermore, in the following, we explain the architecture.

The `DropSpacePin` class is responsible for the user interactions related to the space pins. This responsibility means that the instance of the `DropSpacePin` class takes care of adding spatial anchors and QR code markers and removing them. To set the position for a new space pin, the user has to do an *air tap*.

The `DropSpacePin` class uses an event from the spatial awareness system triggered when an air tap happens. Once triggered, this class checks if the hit object is not a user interface element. Unity provides the concepts of layers, one of them being the UI layer, to check which layer an object resides. If an object is not in the UI layer, then the `DropSpacePin` calculates the rotation for the object so that the created space pin points in the direction of the start of the beam. This set rotation makes it easier for the user to position the space pin. A 3D object representing a tunnel chainage sign is the visualization for a space pin. Figure 7.4 shows this 3D object. This rotation is calculated by first calculating the direction to the start of the beam. The calculation of this rotation involves subtracting the start position of the beam, represented in a three-dimensional

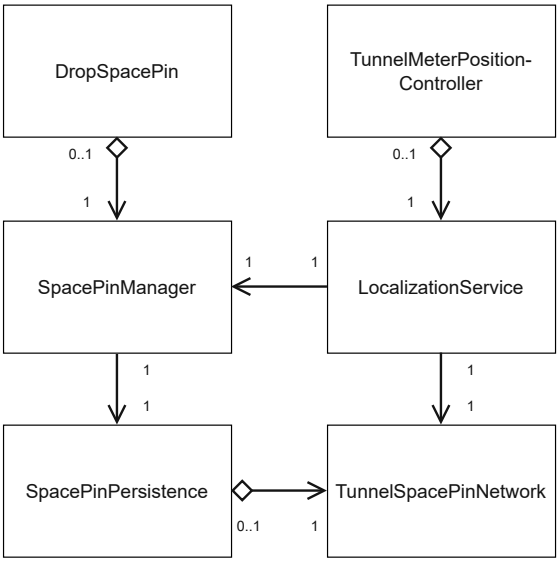

Figure 7.3: A simplified class diagram to represent the architecture for the localization solution.

Figure 7.4: Virtual tunnel chainage sign representing a space pin.

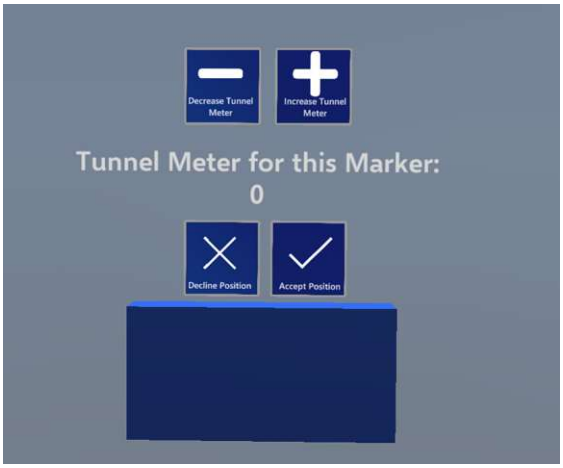

Figure 7.5: Helper tunnel chainage sign to position the space pin and set the tunnel chainage.

vector, from the hit position, i.e., where the beam and the first 3D object intersect. The `DropSpacePin` class then calls the `SpacepinManager` class with the hit position and the calculated rotation to create the actual space pin after this calculation.

A similar approach applies while removing space pins. However, the position is unnecessary since the triggered air tap event already includes the target object intersecting with the beam. The `SpacepinManager` then receives this target object to remove it. The user must click a button to change the mode to differentiate if a regular spatial anchor, a QR code marker, or a deletion should happen. This button directly calls a function in the `DropSpacePin` class to change the mode.

The `SpacepinManager` class creates, loads, and deletes space pins. The functions called by the `DropSpacePin` class to create space pins are `CreateQRSpacePinAtPosition` and `CreateSpacePinAtPosition`. They both create space pins but differ slightly in how the user accepts them before being saved in the WLT framework. The `CreateSpacePinAtPosition` function does not create a space pin itself but shows a helper model for the user interaction. This helper model looks exactly like the virtual tunnel chainage object but includes an estimated distance to the previous space pin and two buttons to set the tunnel chainage for that space pin. Figure 7.5 shows an example of this helper model. This helper model is positioned and rotated with the previously given position and rotation from the `DropSpacePin` object. Additionally, the user can still move and rotate this helper model to the correct position of the real-world tunnel chainage. The user can move this helper by either holding down an air tap on the 3D object or directly grabbing the helper with their hands. To achieve this, the MRTK provides classes that take care of the grabbing and moving part when an object attaches an instance of this class. Once the helper model is positioned, and the real-world tunnel chainage is set, the user can click a button to either accept the space pin or cancel the

whole creation. The accept button calls another function on the SpacepinManager that creates the 3D object, adds an instance of a SpacePin (a class provided by Microsoft), and positions it at the set location from the helper. Lastly, the SpacepinManager calls the SpacePinPersistence with the newly created space pin.

The CreateQRSpacePinAtPosition directly creates the 3D object representing the space pin but does not show the helper model and does not add the SpacePin instance to this object. Instead, an event is triggered that a new QR code space pin was added, which is caught by the QRSpacePinGroup object, which handles the space pin class instantiation and adds it to the 3D object. We used the QRSpacePinGroup class and other related classes from the QR Space Pins sample code [7]. The code from this sample creates space pins at the position of QR codes in the real world. This implementation includes handling the position finding of the QR code and translating it into Unity coordinates. Furthermore, reading the information from the QR code to identify markers again is also included. Moreover, lastly communicating this position to the WLT framework. A few changes were made to the QRSpacePinGroup to retrieve the calls from the SpacepinManager. We also changed the positioning of the created space pin, such that the position is centered on the physical QR Code and not in the corner. The function RemoveSpacePin calls the persistence with the given 3D object, which then removes it from the scene and the space pin collection. Lastly, the SpacepinManager also takes care of two other points. First, it contains two functions to enable and disable the possibility of moving already created space pins. We added this functionality to avoid accidental movement of space pins when the user is not in edit mode. These functions iterate over all available space pins and remove the instance of the class that takes care of moving the object. Second, multiple events are propagated from the SpacePinPersistence, in case space pins were loaded, removed, or created.

The SpacePinPersistence, as the name implies, takes care of saving and holding the space pins. The localization solution stores the space pins in two locations. Firstly, the WLT framework holds the real-world position in relation to the Unity coordinate system. Moreover, the SpacePinPersistence holds information about the space pins, including the tunnel chainage for its position, the position and rotation in Unity's coordinate system, and the name to identify it later. This persistence also includes an instance of the TunnelSpaceNetwork, which is responsible for storing all the active space pins for a specific tunnel. Therefore, all actions, such as creating, removing, and loading space pins, are also reflected in the TunnelSpaceNetwork. As already mentioned, the save function in the SpacePinPersistence takes the position, rotation, name, tunnel chainage, and a flag if it is a QR Code marker and saves it. SpacePin-Persistence stores this information in a JSON file saved on the HoloLens directly. Ideally, the persistence will store this JSON data on a server in the future, so after the setup, every user with a HoloLens can use the same localization. Lastly, Space-PinPersistence calls the WLT framework to save the current world captured. The

[7] https://microsoft.github.io/MixedReality-WorldLockingTools-Samples/Advanced/QRSpacePins/QRSpacePins.html

created file from the WLT framework should also be stored on a server in the future to have all the information needed for the localization solution. There also exists a `LoadCreatedSpacepins` function to load this information again. This function reads the stored JSON files and creates a new space pin for each entry. A 3D model is shown again, such as for manually created space pins. The created space pins must also get the names from the JSON file assigned before calling the WLT framework to load the world again. If the names of the space pins are different from their creation name, then the WLT framework does not know which space pin belongs to which object. Lastly, the `SpacePinPersistence` also contains a function to clear all space pins, as well as delete the data created by the WLT framework.

The `TunnelSpacePinNetwork`, as already briefly noted, contains all space pin references. Space pins are called nodes in the `TunnelSpacePinNetwork` since they serve two purposes. The WLT framework uses the space pins to interpolate Unity's coordinates system with the real world. However, in the `TunnelSpacePinNetwork`, the space pins are used for their position to calculate distances and find locations and neighbors. The space pins reference Unity's Transform class that stores the position, rotation, and scale of the space pin in the scene. A sorted list holds these nodes, i.e., lists that sort themselves when adding new elements. The key to sorting this list is the assigned tunnel chainage by the user. Therefore, a space pin with an assigned tunnel chainage of 0 will be at the beginning of the list. Nodes can be added and removed from the `TunnelSpacePinNetwork`. The distance calculation between the nodes works by utilizing the `Vector3.Distance` function of Unity that returns the distance of two given three-dimensional vectors. For two three-dimensional vectors

$$\boldsymbol{r}_1 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}, \qquad \boldsymbol{r}_2 = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix}, \tag{7.1}$$

one can calculate the distance

$$\mathrm{Dist}(\boldsymbol{r}_1, \boldsymbol{r}_2) = \|\boldsymbol{r}_1 - \boldsymbol{r}_2\| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}. \tag{7.2}$$

This approach is possible because of the shape of a tunnel. Therefore a chain of nodes is created for the localization. However, an exact placement of the space pins is needed to achieve accurate distance results. The start node of this chain has the position of tunnel chainage zero, while the last node has the highest tunnel chainage. Figure 7.6 contains a visualization of this concept of connected nodes. The green lines represent the lines for the calculated distances, and the blue rectangles represent the nodes.

The most crucial function for the localization inside the tunnel is called `GetTunnelMeterForOriginPoint`. This function takes the origin point as a Transform object as an argument to calculate its position. `GetTunnelMeterForOriginPoint` also takes the nearest node to the origin point as an optional argument. This function calculates the nearest node with the `GetNearestNode` function if not supplied. `GetNearestNode` searches for the nearest node to the given position vector. This function uses a variation

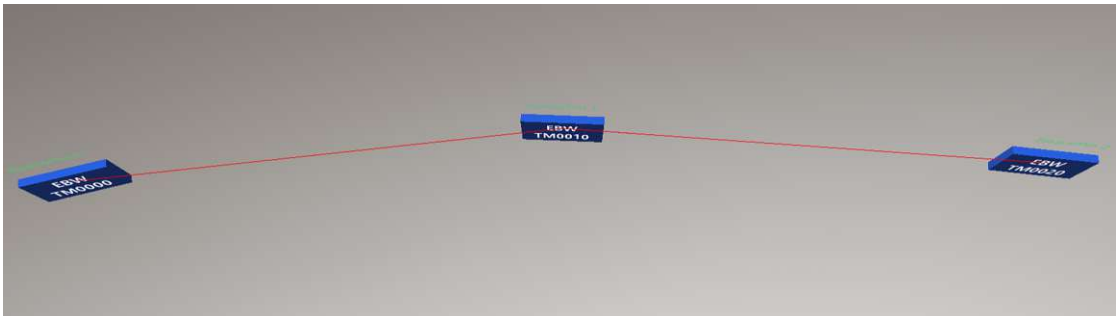

Figure 7.6: The connection between the nodes visualized from a top view perspective.

of the binary search to reduce the time and resources to find the nearest node. We applied this strategy since a tunnel will have many nodes if fully mapped.

Usually, the binary search strategy looks at the middle of a list and checks three cases. Either, the value in the middle of the list is already the searched-for value. If this is the case, the search returns the value. Otherwise, if the selected value is smaller than the searched value, the search function looks in the middle of the right half of the list. Moreover, if the selected value is greater than the searched value, the search function looks in the middle of the left half of the list. This search uses an iterative approach until the function finds the searched-for value or reaches the end of the list. A precondition for this search is a sorted list.

The binary search implementation for `GetNearestNode` does not look for an exact value. Instead, the implementation looks for the smallest distance to the given position. The search uses the `Vector3.Distance` function to compare the distance between the node in the middle, the next node, and the previous node of the list. After that, the function checks if the middle node has a smaller or equal distance to both neighbor nodes. If this is the case, we have already found the nearest node. Otherwise, if the previous node has a smaller distance, then the left half of the list is searched for a smaller distance. Furthermore, the same goes for the next node if it has a smaller distance to the origin point. If the search finds the correct node, the index of that node is returned. With the nearest node found, the `GetTunnelMeterForOriginPoint` function searches for the neighbor nodes for the location of the origin point. This process is still needed since the position can be between the nearest node and the next node or the previous node.

The function `GetNeighborsForGivenPosition` takes care of this search. This function gets both neighbors of the nearest node by simply using the previous and subsequent entries in the sorted node list. Furthermore, this function calculates the distance of the origin point between the nearest node and the next node and, respectively, with the previous node. The calculation of these distances returns only a non-zero value for the neighbors surrounding the origin point. Checking if the value is not zero helps return the correct two neighbors of the origin point.

The `GetDistanceForOriginPointBetweenCutEnds` function calculates the dis-

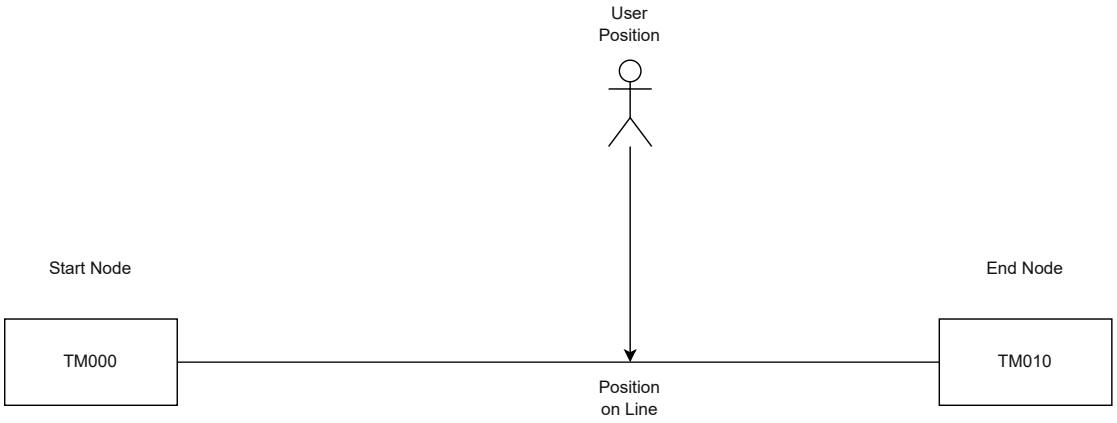

Figure 7.7: Positioning on the line between two nodes.

tance between two nodes. This function first searches for the closest point on the line between the two neighbor nodes. Figure 7.7 illustrates this position between two nodes. This position is needed because the position given is likely not directly between the nodes. They are likely not in between the nodes since the nodes (space pins) exist only on the tunnel lining. Therefore, this closest point is needed to calculate the distance on the line between the nodes. The function that calculates this position is called `GetClosestPointOnFiniteLine`. `GetClosestPointOnFiniteLine` uses a vector projection to calculate the position. The following shows the exact calculation.

One can calculate the direction $\boldsymbol{r}$ from the start point of the line $\boldsymbol{s}$ to the end point $\boldsymbol{e}$ by

$$\boldsymbol{r} = \boldsymbol{e} - \boldsymbol{s} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{7.3}$$

$\boldsymbol{e}$ represents the next node after the origin point $\boldsymbol{o}$, while $\boldsymbol{s}$ represents the previous node. The line length $\|\boldsymbol{r}\|$ results from the magnitude calculation of $\boldsymbol{r}$ by

$$\|\boldsymbol{r}\| = \sqrt{x^2 + y^2 + z^2}. \tag{7.4}$$

This magnitude represents the length from the start to the end node. The normalized vector $\hat{\boldsymbol{r}}$ results from

$$\hat{\boldsymbol{r}} = \frac{\boldsymbol{r}}{\|\boldsymbol{r}\|}. \tag{7.5}$$

One can calculate the length p from the start of the line between the start and end node by

$$p = (\boldsymbol{o} - \boldsymbol{s}) \cdot \hat{\boldsymbol{r}}. \tag{7.6}$$

Lastly, one can calculate the vector $\boldsymbol{g}$ located on the line by

$$\boldsymbol{g} = \boldsymbol{s} + \hat{\boldsymbol{r}}p. \tag{7.7}$$

This line position is then used in `GetDistanceForOriginPointBetweenCutEnds` to calculate the distance between this and the start node. The function then checks if the distance is the same length as the distance between the start and end nodes. If this is the case, the function returns 0 since the origin point is not between the two nodes. After finding the neighbor nodes for the origin point, the `GetTunnelMeterForOriginPoint` function calculates the actual tunnel chainage. This calculation happens in `GetDistanceForOriginPointBetween`, which uses the two neighbor nodes and the origin point. This function is very similar to `GetDistanceForOriginPointBetweenCutEnds` but does not return 0 if the distance has the same length as the whole distance between the start and end node. Additionally, `GetDistanceForOriginPointBetween` uses linear interpolation to get a more accurate result of the calculated location from Unity's coordinates system to the real-world positions.

The following shows this interpolation calculation. One can calculate the percentage of the distance d_p from $\|\boldsymbol{r}\|$, by

$$d_p = \frac{n}{\|\boldsymbol{r}\|}. \tag{7.8}$$

n is the distance from the start node to the position $\boldsymbol{g}$ calculated by `GetClosestPointOnFiniteLine`. $\|\boldsymbol{r}\|$ is the distance from the start node to the end node. The actual tunnel chainage m_f results then from

$$m_f = m_s + (m_e - m_s)d. \tag{7.9}$$

m_e and m_s represent the tunnel chainage values assigned by the user to the start and end nodes respectively. These equations take the percentage of the distance in Unity's coordinate system and use this percentage in the interpolation between the actual tunnel chainages of the nodes assigned by the users. This interpolation should minimize the localization error from Unity's coordinate system to the real-world positioning.

The `LocalizationService` is a service that is responsible for all localization operations and works closely with `SpacepinManager` and `TunnelSpacePinNetwork`. `CheckNodeLoop` is an essential part of the `LocalizationService` and checks the user's current position. This loop executes every second and has two tasks. Firstly, the nearest node is calculated by calling the `TunnelSpacePinNetwork` and stored in a variable. Secondly, the current tunnel chainage for the main camera (which is the user's position) is calculated by calling `GetTunnelMeterForOriginPoint`. An event then publishes the current tunnel chainage of the user position to which other controllers and services can subscribe. Another part of the `LocalizationService` is to expose functionalities from the `TunnelSpacePinNetwork`, such as retrieving positions, nodes, and distances.

The `TunnelMeterPositionController` is a simple script that updates a text label in the scene. This text label shows the user the current tunnel chainage of their position. The controller subscribes to the `LocalizationService` to get this information.

7.1.1 User Interface

The user interface for the localization prototypes went through multiple iterations. Figure 7.8 shows the main window for the marker creation. This menu follows the user around and is directly in front of the user's view. The menu is placed at a distance so the user can directly interact with the buttons by pressing them with their fingers. The name for this type of menu is a near-interaction window. This type of menu contrasts with menus designed for far interaction, which utilize the air tap. The name for windows using an air tap for the interaction is far-interaction windows. Field tests at the tunnel affected the decision for the menu design.

In Figure 7.8, we can see the current tunnel chainage position at the top, which the `TunnelMeterPositionController` sets. We can also see the buttons for *Add Marker* and *Add QR Marker*, which call the previously explained functions `CreateSpacePinAtPosition` and `CreateQRSpacePinAtPosition`, respectively. The button *Cancel Last Marker* cancels a created but not yet accepted space pin through the `SpacepinManager`. *Remove Marker*, as the name suggests, removes an already created marker. This button calls the remove function on the `DropSpacePin` object. The application closes when a user presses the *Close Application* button. Lastly, the *Settings* button opens the settings menu. Figure 7.9 shows this settings menu. In the figure, we can see multiple options related to the localization. It also shows buttons added later for the other prototypes which use localization. The *Save Markers* saves all created space pins, as explained previously through the `SpacepinManager`. Pressing the *Reset Markers*

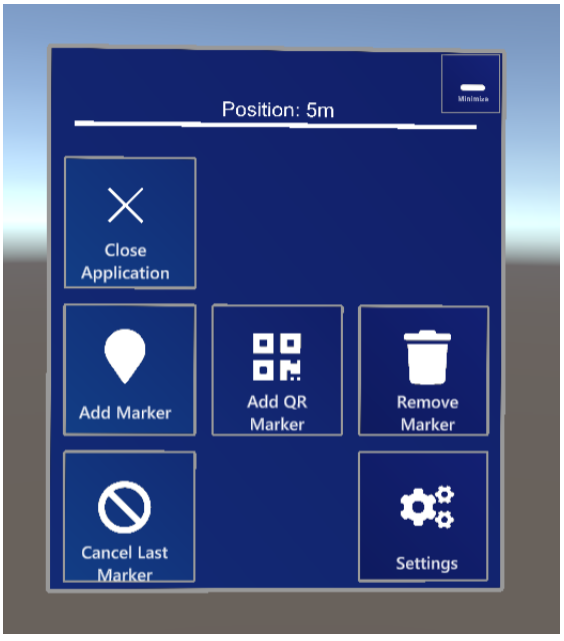

Figure 7.8: The menu to set up the localization for the prototypes.

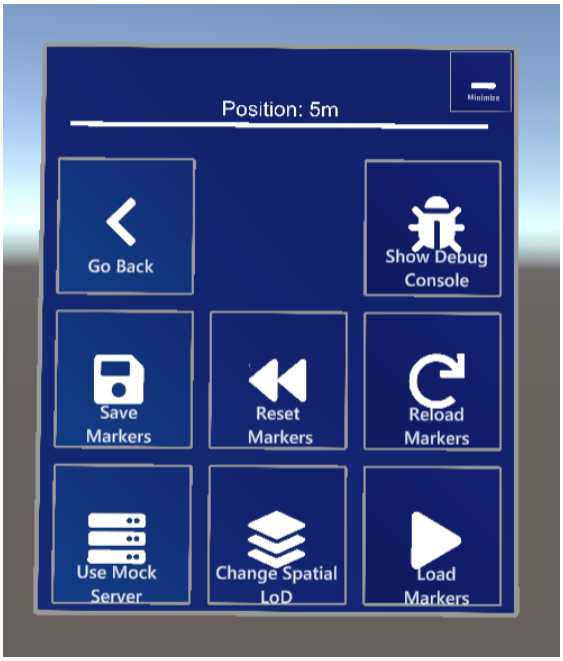

Figure 7.9: The settings menu of the prototypes.

button removes all space pins and deletes the file that holds the WLT information. This button also calls a function for this behavior in the `SpacepinManager`. The *Reload Markers* button reloads the information from the WLT framework and loads the space pins from the JSON file. *Load Markers* is similar to the previous button but loads the space pins and not the whole WLT information again. *Change Spatial LoD* changes the level of detail for spatial mesh detection (spatial awareness system). Spatial mesh detection is responsible for creating meshes, e.g., for the walls, so that space pins can be placed directly on these meshes. The options for the LoD are medium and high since walls in a tunnel are usually very high and therefore need more computing power for the detection. Lastly, the *Show Debug Console* shows a debug console to the user to see the log information created from the application.

7.1.2 Field Tests

We conducted multiple field tests to test whether the localization inside the tunnel works. The following summarizes the tests and results.

Basic Tunnel Field Test

The goal of this field test was to answer the following questions.

1. How well does the WLT approach work in a tunnel?

Figure 7.10: The EBW tunnel environment.

2. How well does the creation and loading of space pins work in a tunnel?

3. How well does the creation and loading of space pins in the form of QR codes work in a tunnel?

4. Does the distance calculation between two tunnel chainage signs show the correct distance of 10 m (given the value is rounded to one decimal)?

We conducted the tests in two of the tunnels at the ZAB. One of the tunnels has no inner lining but an auxiliary shotcrete lining and a flat floor made of asphalt without distinct features. This tunnel is called EBW. Figure 7.10 shows a picture of this environment. We measured the illuminance with the sensors of a smartphone. An illuminance between 6 and 15 lx was measured.

The second tunnel consists of an outer lining and railway tracks. This tunnel is called EBO. Figure 7.11 contains a picture of this environment. We measured an illuminance between 43 and 162 lx with an average of 70 lx for the EBO tunnel.

We conducted multiple tests to answer the previous questions. First, two space pins were created by placing them directly on two consecutive tunnel chainage signs. After that, we saved the space pins. We also restarted the HoloLens to check if the space pins stayed

Figure 7.11: The EBO tunnel environment.

at the correct location. Following the restart, we reloaded the space pins. We repeated the same steps for the tests with the QR code space pins. Furthermore, we repeated both tests in both tunnels.

The tests revealed that the WLT also works in a tunnel environment. We activated the debug mode of the WLT anchors that visualized the automatically created anchors. These anchors stayed the same after the restart of the HoloLens. The space pin loading for both approaches also worked without problems in both tunnels. However, one problem arose with spatial mesh detection. After a few tests, the spatial mesh detection did no longer recognize the walls. No space pin could be placed after this since the creation depends on the wall mesh detection. After restarting the device, it worked again but consistently stopped detecting the walls after a short time.

To solve the previous issue, we researched in forums and read through the documentation of Microsoft. This research revealed that the spatial awareness settings have to be changed. A property called *level of detail* has to be set to medium or high to detect more enormous walls according to Microsoft[8]. The higher the level of detail, the more it impacts the application performance. However, the space pin placement depends on this

[8] https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/spatial-awareness/configuring-spatial-awareness-mesh-observer?view=mrtkunity-2022-05

computation. The level of detail is therefore only set to high or medium when needed and otherwise completely disabled to save on resources.

Distance Calculation and Usability Tunnel Field Test

We revisited the same tunnels for this field test as in the previous one. The goal of this field test was to answer the following questions.

1. Is the accuracy of the localization between space pins 20 m apart still acceptable? Acceptable is a maximum of 20 cm over the tunnel chainage.
2. Are there problems when other users try to place the space pins?
3. Are there problems with the interaction of the menus displayed?
4. Are there features missing in the menu for the user?

We checked the accuracy between three tunnel chainage signs, i.e., on a length of 20 m. In this section, we drew multiple lines with chalk on the ground. The lines had a distance of 1 m. We mentioned this previously in Chapter 2. Furthermore, Figure 2.2 depicts this process. After that, we walked along the section and checked if the tunnel chainage displayed in the UI showed the correct information as long as the position of the HoloLens was between two lines. After explaining the process, we let users test the space pin creation.

From the tests, we gathered that the accuracy is sufficient for determining the tunnel chainage from the user's position. The text label changed to the next tunnel chainage, even when the user took a small step forward such that the HoloLens hovers a few centimeters over the chalk line. A problem arose when the users tested the space pin creation. It took a small amount of time for the user to use the air tap. However, in the end, all users managed to place space pins. Another problem observed during the test was connected to the WLT. After a user minimized the prototype application on the HoloLens and opened the application again, the WLT anchors rotated such that the space pins no longer aligned with the tunnel chainage signs, and the localization also no longer worked. A restart solved this problem again. Another solution to this problem could be using QR codes which were more reliable for fixing the orientation of Unity's coordinate system through WLT to realign the anchors with the real world. Lastly, we found that the menu design at the time of the tests did lack some features the users demanded. The following summarizes these features.

1. The users wanted a button to close the application.
2. They also wanted a button to reset the WLT and space pin information.
3. Another request was for a hint for the placement of space pins that explains the action users have to perform.

4. Lastly, the users wished for a more streamlined design for the menus. For example, using the same UI elements and grouping buttons into different menus.

7.2 Acceptance Inspection Prototype

In Figure 7.12, we can see how to use the acceptance inspection prototype during the inspection. First, the user has to finish the localization setup if this is still needed. After that, the user can start a new inspection. During the inspection, the site supervision must compare the stored information for the round with the actual work. Therefore, the user can open a window with all information from TIMS for the specific round of their location. Examples of important information displayed in this window are the built-in support measures. Another button allows documenting, for example, the found support measures in the round information window. Users are guided through the following process when they press this button. The user receives a prompt to use an air tap on the place in the tunnel to document it (e.g., a spot on the tunnel wall). Afterward, the solution captures a photo of the spot under investigation. After the user has accepted the captured photo, they can add a written note with a virtual keyboard or voice commands. Lastly, the documented object is stored in the current report if the note is accepted. The user repeats this process for the current round until they document all support measures or spots. The user can then include the current round in the report to mark it as inspected. After the inspection of the last round, the user can click the *finish report* button to save the report.

The implementation of this prototype utilizes the localization prototype to get the position of the user. In Figure 7.13, we can see the three-tier architecture of the prototype. We omitted all classes explained earlier or those not needed to understand the implementation from the figure.

The class that handles the interaction with the user for the inspection is called `Inspection-`

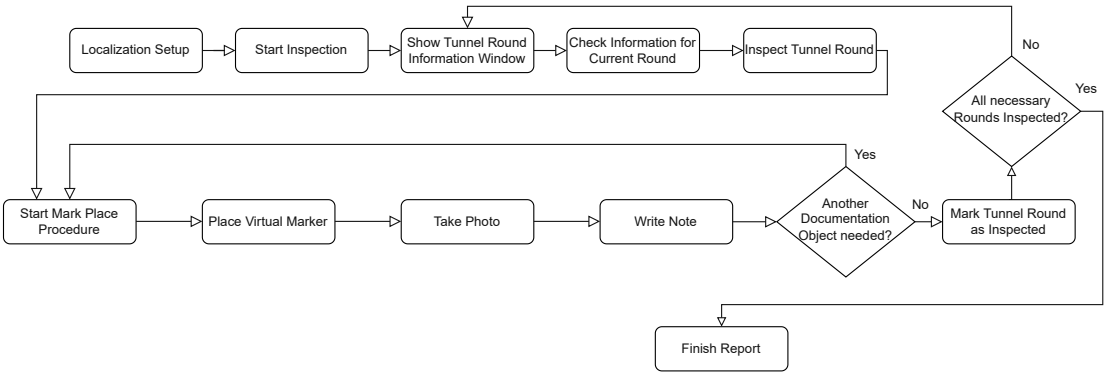

Figure 7.12: The process of the Acceptance Inspection Prototype in the form of a flowchart.

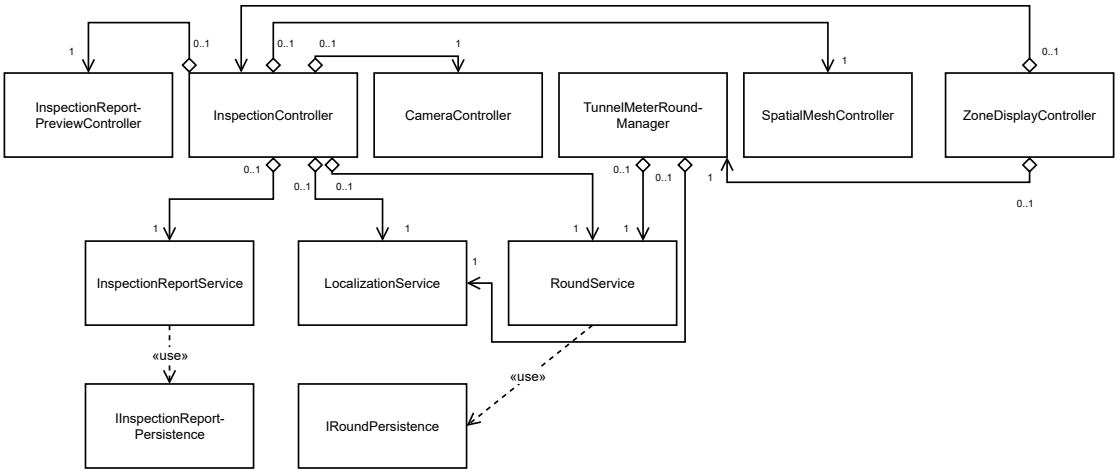

Figure 7.13: A simplified class diagram of the Acceptance Inspection Prototype.

`Controller`. As seen above, the first interaction with the prototype is to start an inspection. When clicking the start inspection button, it triggers an inspection through the call to the function `StartInspection`. `StartInspection` calls the `InspectionReportService` to start a new report.

The `InspectionReport` is the object that represents the information created during the inspection. The `InspectionReportService` holds the `InspectionReport` instance. This service takes care of adding and removing rounds and documentation objects. The `InspectionReportService` provides a function to check if a round already exists in the report. The round information window button needs this check to determine if the user has already inspected a round. Lastly, this service provides the function `SaveCurrentReport`. This function takes the current `InspectionReport` object and forwards it to the `IInspectionReportPersistence`. `IInspectionReportPersistence` is an interface with one implementation to store the report as an HTML file. At a later stage, a new `IInspectionReportPersistence` implementation will store the reports on the TIMS server.

After the start of an inspection, the user can show the tunnel round information window. The `ShowRoundInfoWindow` function opens this window. This window contains a reference to the `ZoneDisplayController`. The `ZoneDisplayController` shows the information for the current round for the users' location. `TunnelMeterRoundManager` provides this information through the event `OnCurrentRoundsRetrieved`.

The `TunnelMeterRoundManager` class keeps track of the current round for the users' location. The `OnUserPositionChanged` event exposed from the `LocalizationService` continuously updates this information when it finds a new tunnel chainage position for the user. The `LocalizationService` then sends this tunnel chainage to the `RoundService` (through that event), which talks directly to the `IRoundPersistence`.

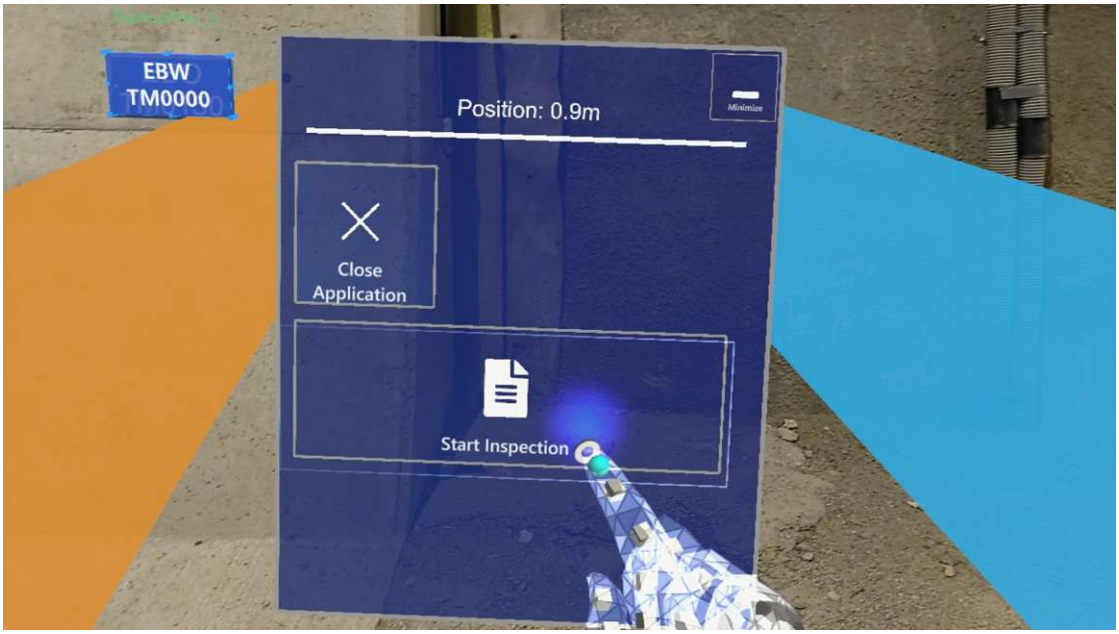

Figure 7.14: The borders that represent the start and end of a tunnel round.

`TunnelMeterRoundPersistence` implements `IRoundPersistence`, which talks to the TIMS server. `GetRoundsForTunnelMeter` is an essential function in this class and retrieves all rounds for the given tunnel chainage. This function sends a query to the TIMS REST API to get all rounds for which the start position is less or equal to the tunnel chainage. Furthermore, the end position needs to be greater than the tunnel chainage. The persistence also includes a function to get a specific round by its id.

The `TunnelMeterRoundManager` also has a second function. Some users mentioned during the tunnel field tests that visualizing the start and end of a tunnel round could help during the inspection. This manager implements this visualization with the `CreateTunnelRoundBorders` function. `CreateTunnelRoundBorders` creates two cuboids 12 m long, 1 cm thick, and 1 m high. The 12 m are important so that the rectangle intersects with both tunnel walls. These rectangles represent the start and end of a tunnel round and serve as borders. Figure 7.14 depicts the orange and blue rectangles that represent the borders. `CreateTunnelRoundBorders` uses the start and end position of the currently held round object, i.e., the round the user is standing in.

We had to extend `TunnelSpacePinNetwork` with a reverse lookup of a tunnel chainage. This function is called `GetPositionOfTunnelMeter`. `GetPositionOfTunnelMeter` takes a tunnel chainage as an argument and returns a three-dimensional vector with the position in Unity's coordinate system. The reverse lookup first searches for the nearest node behind the tunnel chainage's position. For example, if the location of a node is at tunnel chainage 10 and the next node's position is at meter 20. Moreover, if the specified tunnel chainage is 12, then the node at meter 10 is the nearest before. Another

binary search implementation achieves the finding of this node. This time, however, the binary search returns the node's index when the assigned tunnel chainage is less or equal to the searched tunnel chainage. Another constraint is that the next node's assigned tunnel chainage has to be greater than the searched-for one. `GetPositionOfTunnelMeter` then does the following calculation. First, this function calculates the distance m_n from the start node tunnel chainage m_s to the searched-for tunnel chainage m_f by

$$m_n = m_f - m_s. \tag{7.10}$$

Afterward, `GetPositionOfTunnelMeter` calculates the direction $\boldsymbol{r}$ from the end node $\boldsymbol{e}$ to the start node $\boldsymbol{s}$ by

$$\boldsymbol{r} = \boldsymbol{e} - \boldsymbol{s}. \tag{7.11}$$

The distance w according to the real world, i.e., the assigned tunnel chainages for the nodes, results from

$$w = m_e - m_s. \tag{7.12}$$

m_e is the tunnel chainage for the end node. We then calculate the distance d of the start node $\boldsymbol{s}$ and end node $\boldsymbol{e}$ in Unity's coordinate system by

$$d = \text{Dist}(\boldsymbol{s}, \boldsymbol{e}). \tag{7.13}$$

An inverse linear interpolation calculates the percentage of the distance d_p between zero and w by

$$d_p = \frac{(m_n - 0)}{(w - 0)}. \tag{7.14}$$

This inverse linear interpolation is needed to reverse the interpolation done during the creation of the nodes. d_p is then used to calculate the fraction of d between the two nodes in Unity's coordinate system that produces the distance d_u in Unity's coordinate system. We calculate this by

$$d_u = dd_p. \tag{7.15}$$

Lastly, the position returned from the `GetPositionOfTunnelMeter` function is $\boldsymbol{h}$ calculated by the simple vector addition

$$\boldsymbol{h} = \boldsymbol{s} + (\hat{\boldsymbol{r}} d_u). \tag{7.16}$$

The `CreateTunnelRoundBorders` uses the position calculated by `GetPositionOfTunnelMeter` to place the borders respectively. The function `GetNeighborsForGivenPosition` retrieves the node behind and in front of the border. These neighbors are then used to calculate the orientation of the borders since a tunnel may have curves that would make the borders no longer usable if they had a fixed orientation. The rotation calculation happens by subtracting the next node from the previous node to get the direction between the nodes. Afterward, the function `Quaternion.LookRotation` provided by Unity is used to get the rotation for a provided forward and upwards

direction. This function, therefore, calls `LookRotation` with the calculated direction as the forward direction and the upwards direction with the up vector $(0, 1, 0)$.

A reference to the current tunnel round object is stored in the `ZoneDisplayController` once the `OnCurrentRoundsRetrieved` is triggered. However, this controller only shows the newest round information with the `UpdateCurrentRound` function. The user can decide when the new round information is visible. A button on the tunnel round information window triggers this function when pressed.

Another important function is called `MarkPlace`. This function calls the `SpatialMeshController` to enable spatial mesh detection. The mentioned `SpatialMeshController` is a class that provides basic functionality affecting the spatial awareness system. This controller can enable, disable and set the level of detail of the spatial mesh detection.

`MarkPlace` also sets a flag that the user is now creating a marker for a documentation object. When this flag is active, an air tap can trigger the `OnPointerClicked` event. This air tap places a marker at the intersection of the air tap and the wall, similar to the space pin placement. The `OnPointerClicked` function enables then the `PhotoDialogController` object that has a reference to the `CameraController`. After the user accepts the dialog provided by the controller, the controller calls the `CameraController` to take a photo. The `InspectionController` subscribes to the `CameraController`'s `OnPhotoAccepted` event, which triggers when taking a picture or canceling the process. If the `OnPhotoAccepted` event is triggered, then `CameraManager_OnPhotoAccepted` is called. This function opens a text box to which the user can add text. Additionally, it calls the method `Open` on `TouchScreenKeyboard` to show the virtual keyboard provided by Microsoft. The user then has to accept or decline the added text. Afterward, this function calls the `OnNoteEditFinished` event if the user accepts the text. This function first retrieves the tunnel chainage for the place that was marked. A call to `GetTunnelMeterForOriginPoint` on the `LocationService` object returns this result. The `InspectionController` calls the `RoundService` to retrieve the round for the marked place. The function `GetRoundForTunnelMeter` provides this information for the provided tunnel chainage.

With this information, the `InspectionController` creates an `InspectionObject` that includes the note, photo, location, and round for the marked place. The function `AddInspectionObjectToReport` in the `InspectionReportService` then adds it to the report. Another vital function is `FinishInspection` in the `InspectionController`. `FinishInspection` enables the dialog that shows the report preview. This function also calls the `PreviewReport` function on the `InspectionReportPreviewController` with the current report object from the `InspectionReportService`. The `InspectionReportPreviewController` prepares the window to show the round and `DocumentationObjects` created during the report. The window that references the `InspectionPreviewController` has a separate button that calls the `InspectionReportService` to persist the report.

7.2.1 User Interface

This section shows the implemented user interfaces for the acceptance inspection prototype.

The first menu that we added for this prototype was the hand menu. This menu opens when the user holds a flat hand in front of the HoloLens. Figure 7.15 shows this menu. We created the hand menu to switch between the localization setup mode and a mode for the use case only. These modes are named build mode and user mode, respectively. Additionally, we added another button to open the main menu. This button is needed since the users can minimize the main menu to not be in the way of the interactions. When the site supervision switches to the user mode and has not started an inspection yet, the menu in Figure 7.16 is displayed. This menu only allows the user to start an inspection and close the application. At the top of the menu is a label showing the current tunnel chainage. We decided to keep this information for all main menus, i.e., the menus that the hand menu can open. The active inspection menu is displayed if an inspection is active, as seen in Figure 7.17. This menu allows a user to mark a problem, show the information for the current round, cancel the inspection, and finish the inspection, as already explained previously. Figure 7.18 shows the design of the tunnel round information window. This window depicts the information for a tunnel round retrieved from the TIMS server. The tunnel round information window also contains a button to include the round in the report. Including a round in the report means that the user has already inspected this round. The *Update Round Info* button allows the user to show the current round info if the user walked to a location corresponding to another round. Lastly, Figure 7.19 shows the preview window for the inspection report. This window shows the information provided in the saved report. The user can close this window and continue the inspection, i.e., add additional information or save the report to finish the inspection.

Figure 7.15: The hand menu used in both prototypes.

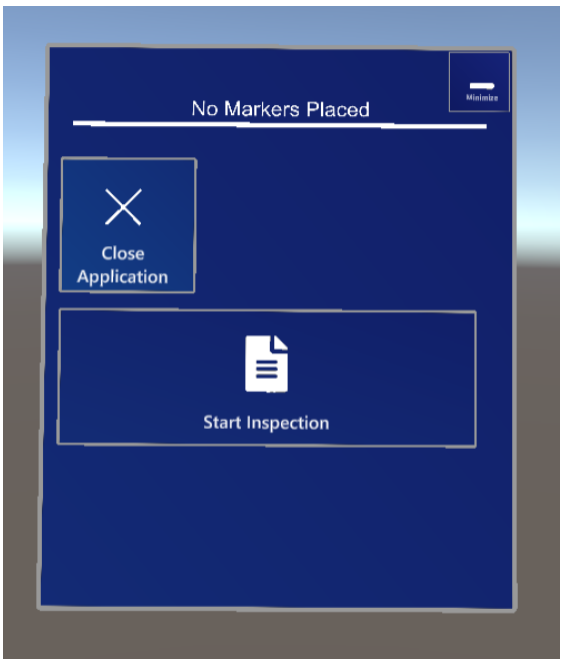

Figure 7.16: The inspection start menu.

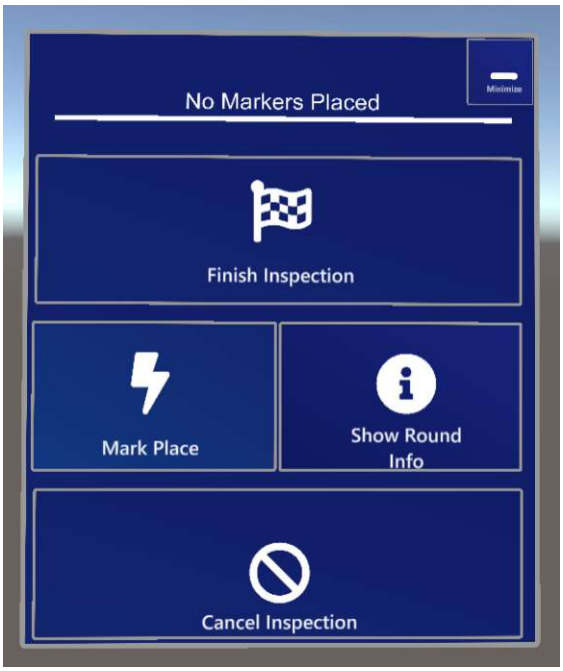

Figure 7.17: The menu that is displayed if an inspection is active.

Figure 7.18: The tunnel round information window.

Figure 7.19: The inspection report preview window.

7.3 Tunnel Face Mapping Prototype

In Figure 7.20, we can see how to use the tunnel face mapping prototype to help geologists map tunnel faces. The first step is the same as in the acceptance inspection prototype. This localization is needed to assign the tunnel face mapping to the corresponding tunnel round. After this, the user can start a new tunnel face mapping. Before annotating, the user has to first span a two-dimensional canvas over the tunnel face. Figure 7.21

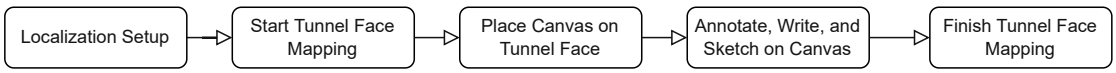

Figure 7.20: The process of the Tunnel Face Mapping Prototype in the form of a flow diagram.

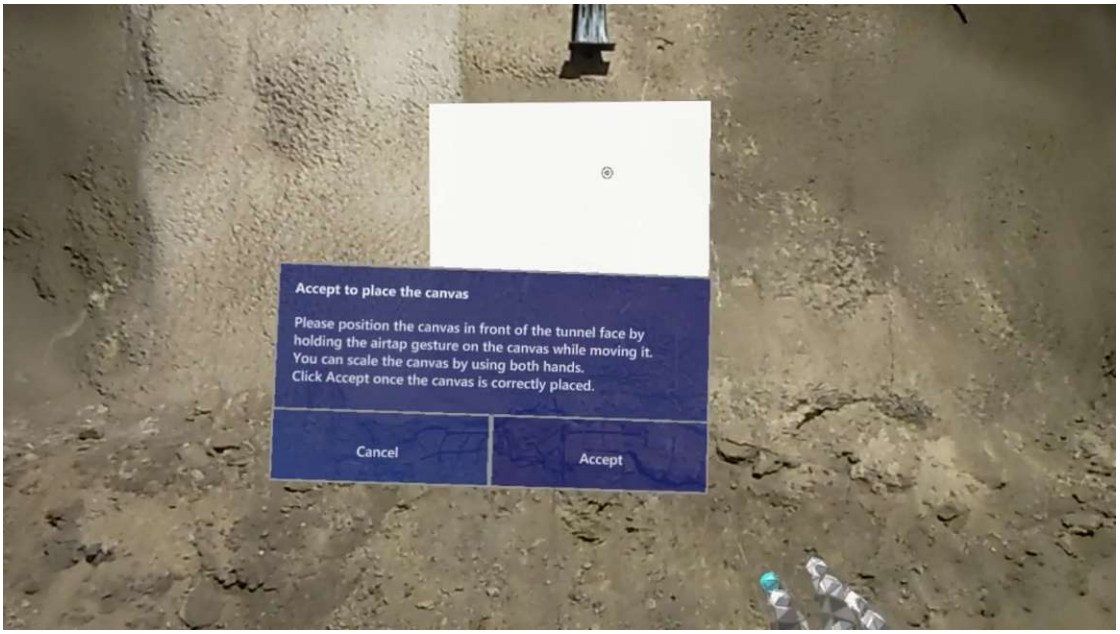

Figure 7.21: The canvas placed for the tunnel face mapping.

shows an example of this canvas. A user can position this canvas by using an air tap on the tunnel face, similar to the placement of space pins. After the canvas is visible, the user has to position, rotate and scale the canvas over the tunnel face. The user can position and rotate the canvas by holding one air tap and pointing at it while moving it. To scale the canvas up, the user has to hold two air taps on the canvas and move both hands away from each other. The canvas becomes smaller if the user moves both hands to each other. The user then has to accept the canvas placement. After that, the canvas becomes transparent with a white border. This transparent canvas is a window on which the users can apply annotations. Annotations can be made in the form of free drawings, adding text, and drawing polygons. The user can finish the tunnel face mapping by clicking a button. This process captures a photo of the canvas with the tunnel face in the background. The acceptance of this photo then completes the tunnel face mapping.

Implementing the tunnel face mapping prototype also builds on the localization prototype. In Figure 7.22, we can see the three-tier architecture of the implementation.

According to the flow diagram in Figure 7.20, the first class is the `CanvasPlacementManager`. This manager listens for the air tap click event triggered if a tunnel face mapping is active. The `CanvasPlacementManager` also calls the `SpatialMeshController` to enable spatial mesh detection during canvas placement. On click, the manager places the canvas object at the intersection of the air tap and the mesh targeted. The manager also rotates the canvas towards the user, so it only needs to be scaled to work with it. The rotation towards the user is calculated similarly to the space pin placement. Finally, the `CanvasPlacementManager` also shows a dialog to the user

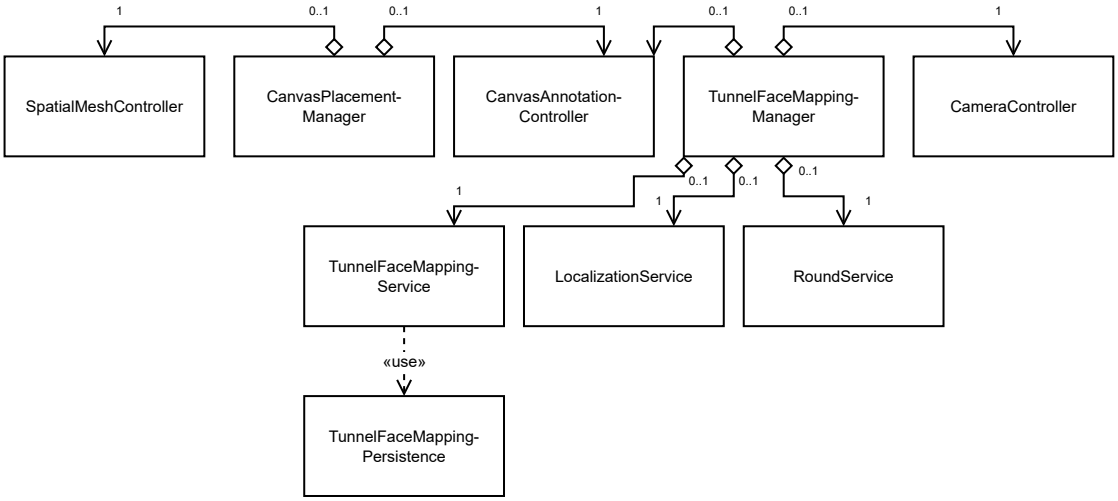

Figure 7.22: A simplified class diagram of the Tunnel Face Mapping Prototype.

to accept the placement. Figure 7.21 displays this dialog. The manager then calls the `CanvasAnnotationController` to activate the annotation process.

The `CanvasAnnotationController` first erases the whole canvas, so it is fully transparent again on activation. Then the borders around the canvas are drawn in white. Lastly, the hand menu, controlling the actions performed in the canvas, is activated. Figure 7.23 shows this hand menu. The `CanvasAnnotationController` has an enum called `DrawMode` that controls the action performed on air tap clicks and air tap drags. An *air tap drag* is an air tap held while being dragged.

For all modes, it is essential first to find the pixel position on the canvas from the intersection of the air tap. We calculate this position by first getting the position of the air tap intersection in the three-dimensional space of Unity's coordinate system. The air tap drag and air tap click event supply this information. After that, we call the function `InverseTransformPoint` on the Transform object of the canvas with the air tap position. This function transforms the air tap position from world space to the local space of the canvas.

The `DrawMode` called *Drawing* means the user can draw freely on the canvas. The `CanvasAnnotationController` listens to the air tap drag event in this mode. We supply the position on the canvas to the `DrawPixelsOnCanvas` once a drag event fires. The drag event fires every frame as long as the user holds an air tap, i.e., has their index finger and thumb pressed together. The free drawing part can be a problem if the HoloLens cannot produce enough frames. Clipped and choppy lines result if the frame rate is too low. To mitigate that, we added a linear interpolation if `DrawPixelsOnCanvas` detects that it was also active in the last frame. `DrawPixelsOnCanvas` applies linear interpolation to the last x position and the current x position. It also applies the interpolation to the last y position and current y position. Furthermore, a

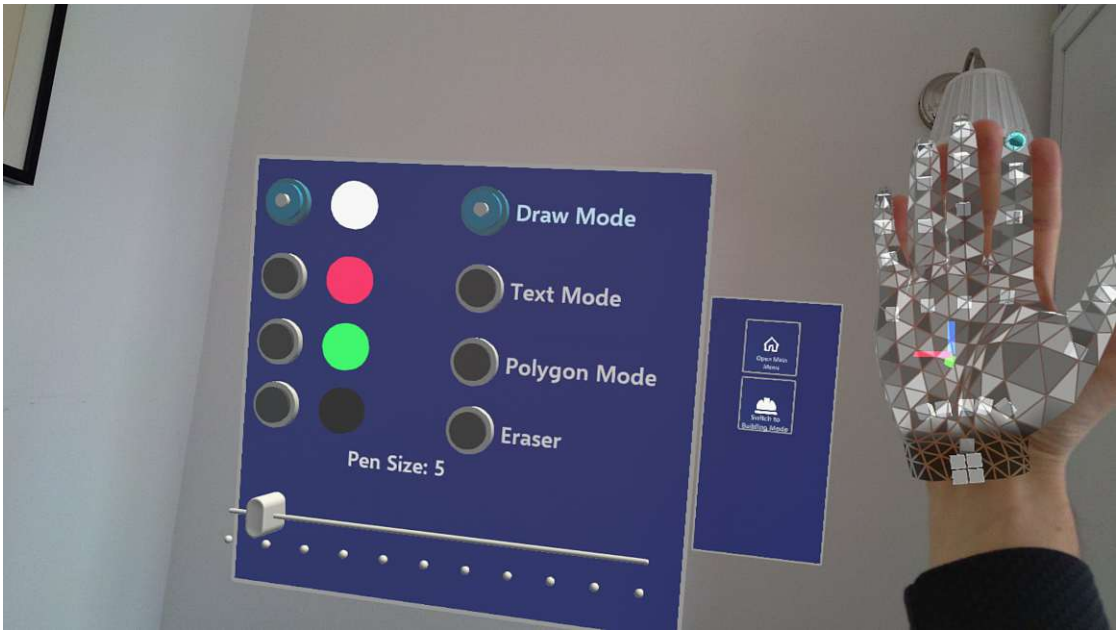

Figure 7.23: The hand menu to control the `CanvasAnnotationController`.

loop supplies the percentage value for the interpolation in 10 % steps from 0 to 100. This loop applies the pixel values directly to the `SetPixels` function on the Texture object of the canvas. `SetPixels` takes the position, size, and colors of the pixels to draw them.

The `DrawMode` called *Erasing* uses the same function as the drawing mode but uses the transparent background color to remove already drawn lines.

The `DrawMode` called *Text* can add text to the canvas. We achieve this by listening to the air tap clicked event. With the position of the air tap intersection of the canvas, we create a `GameObject` that contains a `TextMeshProUGUI` instance. The function responsible also adds this instance of `TextMeshProUGUI` to a list of texts. We then open an instance of `TouchScreenKeyboard` from Microsoft so the user can type in the text. Additionally, we open a dialog to accept the created text. The text mode also works with its enum called `TextMode` to differentiate between adding, editing, and removing text. In the edit mode, we also listen to the air tap click event. We use the same procedure to change the text for adding text. However, we supply the `TextMeshProUGUI` object targeted with the air tap click for this mode. The air tap click event supplies this information if the click targets an existing text element. Lastly, in the remove mode, we remove the element on the canvas and also remove it from the list of texts.

The `DrawMode` called *Polygon* helps the user draw a polygon. To achieve this, the user has to air tap multiple points on the canvas consecutively to draw lines between the points. The user closes the polygon with an air tap on the start point. In the Polygon

mode, we create a visual indication at the first air tap of the user for a new polygon. This visual indication is just a rectangle that has the same size as the current stroke size for the drawing mode. The rectangle is decoupled from the canvas since it is not needed after the polygon is closed. We also draw the first point on the canvas. We create a line from the previous point for the second and every following air tap. For this, we add all air tap positions to a list. Moreover, we use the last two positions in the list for the next line drawing. We extended the `Texture` class by the function `DrawLine` to draw the line between two points. `DrawLine` is an implementation of Bresenham's line algorithm [57] adjusted to work in all quadrants of the canvas.

We found a good explanation in Kaleem, Verma, and Idrisi [58] and will explain this in the following. This algorithm can draw a line on a rasterized image, i.e., find the pixel positions needed to form a line. The algorithm takes two points in the form (x_0, x_0) and (x_1, y_1). The following shows the steps to plot this line (inspired by [58]).

1. First, an initial point has to be selected. We select (x_0, x_0).
2. Second, we have to find $dy = y_1 - y_0$ and $dx = x_1 - x_0$.
3. After that, we get the initial decision parameter P_k with $2dy - dx$.
4. Now we have to check if $P_k < 0$. If this is the case, we plot the next point at $(x_k + 1, y_k)$ and set P_{k+1} to $P_k + 2dy$.
5. If $P_k >= 0$ then the next point will be plotted at $(x_k + 1, y_k + 1)$ and P_{k+1} will be set to $P_k + 2(dy - dx)$.
6. We repeat these steps (4-5) until we arrive at the endpoint (x_1, y_1).

We added to this algorithm an additional check to see which direction the line follows. With this knowledge, we can set the summands in item 4 and item 5 for x_k and y_k. These summands can either be 0, 1, or -1.

In Figure 7.23, we can also see a slider at the bottom of the menu. This slider sets the size for the text and the lines drawn. A simple integer variable holds the value of the slider. The function to draw pixels utilizes the squared value of that variable to set the number of pixels for a given position to be drawn. This variable also sets the size of the text added. We change the text size by directly changing the font size on the `TextMeshProUGUI` object.

The `TunnelFaceMappingManager` finishes a tunnel face mapping after the annotations. `OnPhotoAccepted` from the `CameraController` notifies the `TunnelFaceMappingManager` about this. Before this, the `PhotoDialogController` is shown to the user to take a photo. Once the user accepts the photo, the `PhotoDialogController` calls the function `CameraController_OnPhotoAccepted` of the `TunnelFaceMappingManager`. This function then disables the canvas since the

mapping is no longer active. Additionally, `CameraController_OnPhotoAccepted` calls the function `UploadMapping`. `UploadMapping` takes care of uploading the taken mapping to the persistence. The first step for this procedure is a call to the `LocalizationService` to retrieve the tunnel chainage for the canvas. This information helps retrieve the round for the created mapping. The `RoundService` retrieves the round from the persistence. Afterward, the `RoundService` converts the canvas's photo and texture to a base64 encoded string. The `CanvasAnnotationController` provides the texture from the canvas. The `RoundService` converts the list of texts from the `CanvasAnnotationController` to a list of objects of type `MappingText`. The `MappingText` includes text and its position to save it later in persistence. We pack all the collected information into a `MappingObject`. This object contains the photo, texture, all the `MappingText` elements, and the id of the round. With this information, we call the function `SaveTunnelFaceMapping` on the `TunnelFaceMappingPersistence`. The `TunnelFaceMappingPersistence` is an interface that has two implementations. The first implementation has a connection to the TIMS server to store the information of the `MappingObject` to the belonging round object. However, the interviews revealed that this information should be located on something other than TIMS, e.g., on a server that handles information about tunnel geology. Therefore, we implemented a second `TunnelFaceMappingPersistence` to store the information on the device in addition. This implementation stores all information in a JSON file. In a future implementation, this information should be editable on a computer by supplying not only the photo with the annotations in front of the tunnel face. Instead, the information of the text, including its position and the different lines, should be handled separately.

7.3.1 User Interface

The user interface for the tunnel face mapping prototype also has the streamlined menus already used in the other prototypes. The tunnel face mapping, however, includes only 4 interaction possibilities with these menus.

Figure 7.24 displays the first menu. This menu has only a button to start a new tunnel face mapping.

Figure 7.25 depicts the second menu. This menu gives the user the option to finish an inspection. Clicking the *Finish Tunnel Mapping* button triggers multiple dialogs that guide the user through capturing the photo. When the user clicks on the *Cancel Tunnel Face Mapping* button, the previous menu is visible again, and the canvas for the mapping is hidden.

To better understand how this process looks from the outside, Figure 7.26 shows a user creating a tunnel face mapping with the prototype.

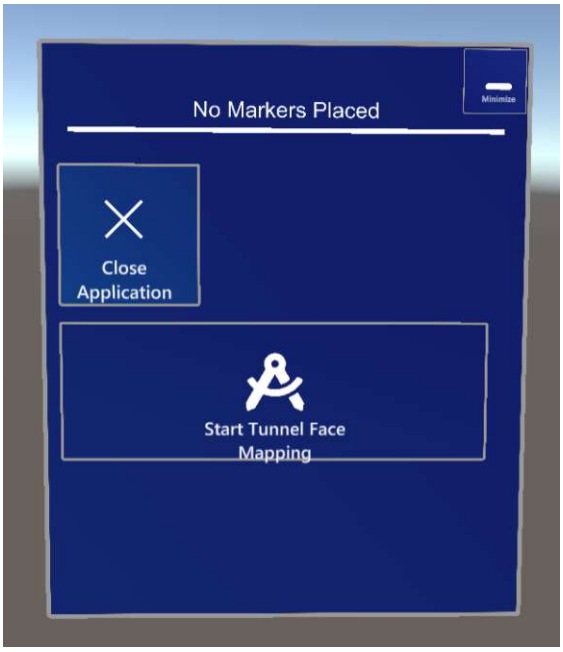

Figure 7.24: The menu that is shown to the user to start a tunnel face mapping.

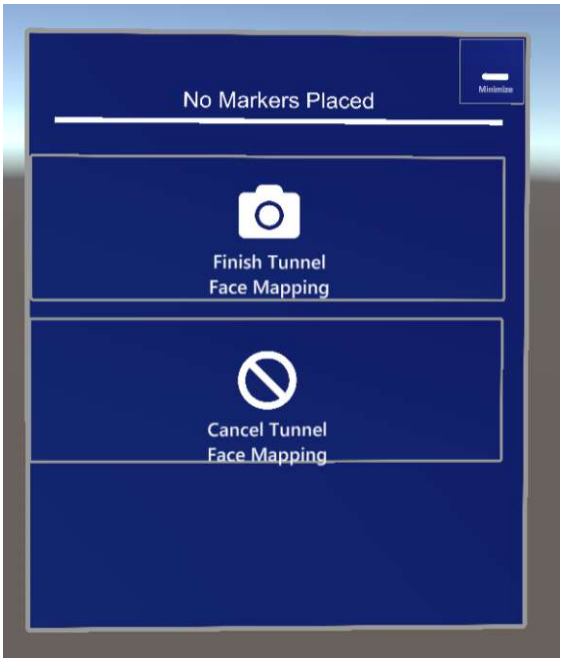

Figure 7.25: The menu that is shown to the user to finish or cancel a tunnel face mapping.

Figure 7.26: A user creating a tunnel face mapping with the prototype.

CHAPTER 8

Results and Discussion

The usability tests happened in a tunnel at the ZAB and involved a convenience sample of five persons. Only two of the participants had previous experience with the HoloLens. All of the participants are involved in or know of tunnel construction.

8.1 Observations During the Usability Tests

The following will summarize the observations during the usability tests for the different prototypes.

Localization

The participants tested the localization setup in a 30 m long section, where the participants had to place four space pins. This task took an average of 3.6 min to complete. Most participants seemed confident during this task. Sometimes the HoloLens did not recognize the air tap, but all participants finished this task without any more significant problems. Moving the space pin did not work for one of the participants. However, they could still add it to the correct position using the air tap click. One of the participants had to restart the process because the accept button was not clicked by accident before the subsequent space pin placement. A participant also tested the prototype with gloves first. This trial revealed that the HoloLens could not recognize hand gestures with gloves. Therefore, we instructed the participants to take off their gloves for the tests.

Tunnel Face Mapping Prototype

The tunnel face mapping prototype was the next prototype tested after the localization. At first, the participants had to place the canvas on the tunnel face. However, none of the participants positioned, scaled, and rotated the canvas perfectly for the prototype. The participants struggled with placing and scaling the canvas since the HoloLens often

did not recognize the air taps. To still test the other interactions with the prototype, we decided to let the users place the canvas as well as possible and then skip to the next part. One of the participants needed help to do this task, so we had to position the canvas for them. The time recorded is, therefore, not meaningful. We also observed that the canvas was sometimes behind the detected meshes of the tunnel face. If the canvas is behind such a mesh, the positioning and drawing do not work since it blocks the air tap. After the canvas placement, the participants had to annotate the canvas. We then asked them to use the different annotation modes (free drawing, text, and polygon). We originally planned that the participants mark specific spots on the tunnel face, but the issues with placing the markers made this impossible. Therefore, the time recording for the tunnel face mapping process is also not meaningful. The participants had to use the keyboard to add text. Three of the five participants struggled with the keyboard but could finally finish the text. All the users (except for one participant who already had experience with the HoloLens) seemed frustrated with the tunnel face mapping setup. The scaling with two hands was the part with which the participants struggled the most. Additionally, we observed that the accept canvas position dialog was often in the way of the users, which made the positioning and scaling harder. The annotation process was more straightforward for the participants, and we observed fewer struggles than previously. However, sometimes the air taps were not recognized on the canvas. This issue happened when the canvas intersected with the tunnel face mesh.

Acceptance Inspection Prototype

The last prototype the participants tested was the acceptance inspection prototype. The first step for the participants was to start an inspection. Then the participants were asked to open the round information window for the respective round. We instructed the participants to go through all visualized rounds in the 30 m mapped during the localization setup. They had to check the information provided in that window for each round they passed. We asked the participants to include the current round in the report if they did not find any anchor as a support measure. This step symbolizes that the user successfully inspected the round. If the participant sees anchors built-in, they had to use the mark place functionality for the number of anchors. For this step, we asked them to mark two places on one lining and one place on the opposite lining. This task took an average of 11.4 min to complete the inspection for 30 m. None of the participants seemed to have issues detecting the built-in support measures. The users seemed very confident about the mark place procedure. Taking the photos was also no problem for the participants. The virtual keyboard part was better for this prototype, and the users seemed to have gotten used to it already. One participant still took longer to enter the text they wanted to add. The virtual keyboard also crashed or closed unexpectedly for one participant. Since the user could not open the keyboard again, the participant had to repeat the whole mark place procedure. We also observed that it needs to be more straightforward for two participants to switch from the round information window to the mark place button. Lastly, the participants had to finish the inspection and preview the report. Two participants struggled with scrolling through the report, which utilizes the

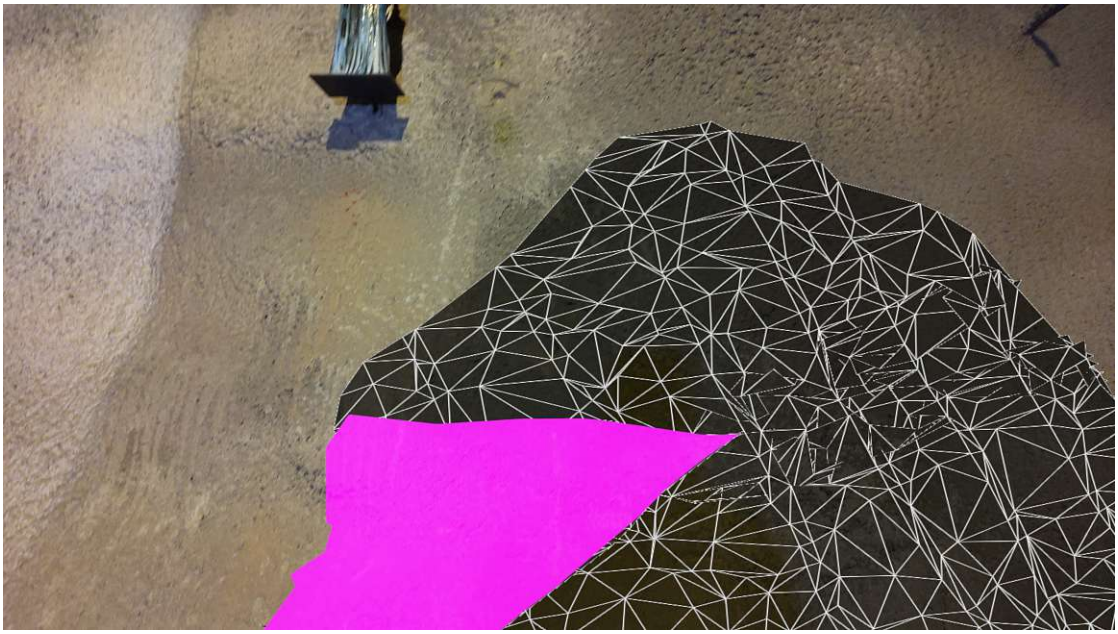

Figure 8.1: An example of the pink artifact that occurred randomly.

air tap hold gesture. Once the users clicked the *Save Report* button, they completed the test. One of the participants could not click the save report button at first. This issue was probably the case since the menu intersected with the wall mesh.

Overall Observations

We observed no more extensive problems with the main menu interactions during the tests. It was clear to all participants how to traverse the menu. Two of the participants needed reminders on how to open the main menu again. The main menu can be opened again through the button on the hand menu. The biggest problems were related to the air tap that needed to be held or dragged. This interaction improved a bit for the third prototype test.

We also observed that the dialogs that should guide the users were sometimes in the way. Three of the participants were hesitant to close the informational dialog windows.

During the usability test, the solution created some pink artifacts. Figure 8.1 illustrates this issue. We think this is a texture error of the white grid displayed from the spatial mesh detection. This issue confused the participants if it occurred since it stayed visible also when the spatial mesh detection was disabled. We also detected that spatial mesh detection sometimes needed a long time to recognize the meshes of the wall. Detecting the meshes of the tunnel face took the longest and only sometimes created satisfying results. The white grid in Figure 8.1 shows that the spatial mesh detection detected only a part of the tunnel face.

Lastly, we used a power bank to charge the HoloLens during the tests since the HoloLens displayed a low battery warning midway through the usability tests. The participants put the power bank in their pockets and used a long cable to connect it to the HoloLens. We did not detect any issues with this approach.

8.2 Evaluation of the Usability of AR in Tunneling

The following summarizes the participants' answers to the questionnaire after the usability tests.

General Interaction with the HoloLens

We gathered from the answers that working with the HoloLens seems intuitive for all participants. One participant mentioned that they had to get used to the restricted field of view but could work with it afterward. Another participant mentioned that windows sometimes obstructed this restricted field of view, which hindered them from doing tasks temporarily.

We found that participants preferred the near-interaction menus, which included buttons they could click directly instead of using an air tap. All participants mentioned some struggles with the air tap to click the buttons on the far-interaction menus.

All participants can imagine wearing the HoloLens for a more extended period. However, one participant mentioned that they would only wear it for 2 hours since they became dizzy after some minutes of usage. One of the participants also mentioned getting eye strain due to the low resolution.

Two of the participants liked the position of the windows. However, two of those and two other participants mentioned that the windows sometimes shifted to places in a different range than the other windows. An example of this is that a window shifted to the ceiling. Two participants mentioned that they would prefer the windows to be farther away. One of those participants suggested the possibility of a dynamic placement by the user in the future.

All participants expressed that the hand and the floating window menu were usable. Three of the participants liked the mix of both menu types. However, one of the participants found the mix confusing and would prefer one menu for all interactions. One participant did not state any opinion about the mix.

Localization Setup

Three participants found the placing of space pins easy and can imagine using this process for longer distances. Another participant said the process was complicated because the user interface was confusing when they accidentally made an air tap, which created a new space pin during the test. We believe this happened because the HoloLens accidentally recognized an air tap when the hand was outside the field of view. However,

this participant thinks this solution is applicable for longer distances with more practice. Another user would prefer a more automated solution.

Two participants found the white grid irritating because it blocked the view of the placement position of a space pin, i.e., the tunnel chainage sign. One participant said a different color with lower saturation would be helpful. The other participants said the grid helped them with the space pin placement. However, one of those participants mentioned that some spots on the tunnel walls were not detected and that a pink artifact irritated them.

Acceptance Inspection Prototype

All of the participants found the acceptance inspection prototype helpful. One of the participants mentioned that the process could be more straightforward by creating reports only for one round instead of merging them. Another participant mentioned that the keyboard input was too slow. This participant would prefer a list of predefined templates to scroll through to speed up this process.

We gathered from the participants' answers that three wished for more detailed guidance through the inspection procedure. One of the participants expressed the need for an initial tutorial on how to interact with the user interface. Two other participants said they would like more information about the next steps in the application. The last two participants found the given instructions through the pop-up dialogs helpful. However, one expressed that these dialogs are distracting if they float in the middle of the view. They would have preferred more subtly embedded guidance hints at the edge of the field of view.

Three participants found the information displayed in the tunnel round information window clear and had no problems obtaining the newest information for the different rounds. One of the participants mentioned that the button labeled *Update Round Info* could be confusing for other users. Lastly, one participant mentioned that they would have liked the information about the support measures to be in the form of tables to separate the information from the rest.

All of the participants preferred this solution compared to a paper-based solution. Two participants mentioned that the feature of photos automatically referencing their location in the tunnel is useful. Another mention by one of the participants was that navigation through the rounds is practical. Furthermore, they found that making information directly available to authorized stakeholders is helpful. Lastly, one of the participants expressed concerns about the environment for the HoloLens. They mentioned that a tunnel construction environment is hot, loud, exceedingly dusty, and wet due to water dripping from the ceiling, and mud and shotcrete could get onto the HoloLens.

Tunnel Face Mapping Prototype

All participants mentioned that positioning and scaling the canvas was difficult. One participant mentioned that placing the canvas with a single air tap was easy. Another participant mentioned that it took a little time to get used to it, but after that, the placement worked better. This participant proposed that a future solution automatically recognizes the tunnel face to create a tunnel-face-shaped canvas.

The annotation part was found intuitive or easy to use by three of the participants. One of the other participants mentioned that they liked the annotation process but needed some time to work with the tools due to gesture detection issues. Another participant also mentioned issues with gesture detection from the HoloLens.

All participants found the hand menu for the tunnel face mapping prototype intuitive and easy to use.

Two participants found the photo capturing easy and had no problems capturing the canvas onto the photo. However, the canvas was not scaled correctly for the participants because of the issues with the air tap recognition. Therefore, capturing the whole canvas in a photo was very easy since it was significantly smaller than the tunnel face. One of the participants asked if they could take the photo before the annotation happened. Another participant mentioned that it would be helpful to show the text in a comment style so the user sees the text once they click a point on the tunnel face mapping. However, this would happen later on a computer, not on the HoloLens. Lastly, one participant noted that this inspection process has time constraints and that selecting text from a predefined list to annotate the tunnel face mapping could improve this. They also mentioned that automatically including the location of the tunnel face mapping is beneficial. Additionally, this participant said a future solution that includes a 3D scan of the tunnel face would be attractive.

8.3 Lessons Learned: Augmented Reality in Tunnel Construction

From the usability tests, we gathered much helpful information about the usability of AR in the environment of tunnel construction. First, we found that near-interaction user interfaces are far easier for participants than far-interaction ones. The issues with the far-interaction menu are especially noticeable when participants lack experience with the HoloLens. We found that participants need significant time to adjust to the interaction with air taps. The amount of time to wear the HoloLens did not seem to be an issue. However, dizziness and eye strain were mentioned already in the short time the user tested the solutions. The users tested the HoloLens on average for 40 min during that day, with short breaks between switching to the other prototypes. We identified that the window placement (distance from the user) was acceptable for some participants. However, a dynamic placement would be a better approach since this depends on the user's preference. Considerations are also needed regarding the low battery life of the

HoloLens when applying it in the field. However, using a power bank seems viable since none of the participants seemed particularly bothered by this approach. Lastly, we found that spatial mesh detection works very well on tunnel lining up to a certain height. However, as in the case of the tunnel face, it took very long to detect spatial meshes. We set the level of detail to medium for spatial mesh detection. Therefore future tests with this level of detail set to high could lead to better results. This also means that the HoloLens needs to use more computing resources to achieve that.

The localization setup approach of placing the markers worked very well for the participants, with only minor issues. Four of the participants can also imagine using this process for longer distances. If we extrapolate the average of 3.6 min for 30 m to 100 m, we get an average of 12 min per 100 m. This value is way below the acceptable 30 min per 100 m. We also are confident that this time will improve when users have more experience with the solution and the HoloLens. The white grid to show the users that a wall mesh was detected led to mixed feelings from the participants. As one participant mentioned, changing the white grid to a more subtle color with less contrast could improve this. We believe this mesh visualization is needed since spatial mesh detection takes time before recognizing the walls. If the users, for example, create markers before the tunnel walls are recognized, it can happen that they will place them behind the wall, which could be irritating for users. This grid approach will no longer be needed if future AR devices can recognize the wall meshes instantly.

Overall the participants found that the acceptance inspection prototype is valuable and would prefer it over a paper-based solution. However, some adjustments are needed for the acceptance inspection to be usable in the field. First, the keyboard is too slow since processes in a tunnel should work as quickly as possible. This process is too slow for tunnel construction since it is expensive and more time leads to higher construction costs. Furthermore, multiple operations are happening simultaneously, which can stall other tasks if the inspection or the tunnel face mapping takes longer than expected. Therefore, as one participant proposed, it would be helpful to include predefined text snippets that can be added to the tunnel face mapping and the acceptance inspection and use the keyboard only sparsely when needed. The users also wished for more guidance through the acceptance inspection solution. A tutorial could be helpful at the first start of the application, as suggested by one of the participants, to explain the menu. Restructuring the user interface could also be helpful to get from the tunnel round information window to the mark place functionality. Adding information text to the tunnel round information window that gives the users hints about the next steps could also be helpful. However, in the interviews, we learned that workers at a tunnel construction site usually know what to do next, and there is no fixed order of steps they have to undertake. These workers sometimes have to change the order of steps depending on the task demands. A tutorial that explains menus and transitions would solve this issue instead of guiding the user through every step all the time. For the tunnel round information window, we found it clear to read, but we need to polish it a bit to separate different kinds of information. The example from one participant to include a table for the support measures seems

viable.

The tunnel face mapping revealed the most problems of the interactions with the HoloLens. Positioning, scaling, and rotating the canvas onto the tunnel face seemed too complicated for the users and needs to be addressed in future versions if used in the field. The users could work tolerably with the single air tap after trial and error. However, holding an air tap was not working for most participants. Therefore, future solutions should automatically detect the tunnel face. This detection could, however, be hard to achieve since the spatial mesh detection did not work well on the whole tunnel face because of its height. An alternative could be for the users to use multiple air taps along the bottom of the tunnel face to outline where and how large the canvas should be. After that, users could set the height with a numerical value so that the canvas spans over the tunnel face. We also found that the dialog to accept the canvas position was often in the way of the user. Therefore, this dialog should be moved outside the center of the user's view or placed on the hand menu. Another issue we found was that the participants could sometimes not annotate the canvas. Future solutions should, therefore, avoid this issue by disabling these meshes after the canvas placement. With this approach, the users can only interact with the canvas. The annotation part was generally more accessible for the participants to use when the air taps were recognized correctly. The hand menu, including the tools for the annotation, was also seen as intuitive by the participants.

8.3.1 Considerations for Further Development

The following will list points that should aid in designing future AR applications. Some of the points are regarding general AR development and testing their usability. Other points are specifically for the use case of AR in the environment of tunnel construction.

1. Using near-interaction menus is easier to understand and use compared to far-interaction menus.

2. Sparing use of air taps is preferable. This interaction type is better suited for advanced users. They provide, however, a good way to place objects onto intersecting meshes such as walls.

3. Using a hand menu to toggle floating windows is a good approach to keep the field of view clear when the user does not need a menu.

4. Floating windows should be dynamically movable since users prefer different distances.

5. Users need an adjustment time before confidently working with the HoloLens. Researchers should consider this in usability tests to gain more insights into the solutions' usability to separate them from the issues with the HoloLens. We propose practice runs utilizing the different interactions of the HoloLens (e.g., air tap, near-interaction tapping).

6. The spatial mesh detection should be disabled in Unity if not used. Additionally, the meshes created during the detection stay active if the detection is disabled. Therefore, this mesh should be disabled if not needed since it can interfere with the user interface.

7. Users need some time to get used to the virtual keyboard. Creating text with a keyboard is, in general, a slow process. Speech recognition works much better, which the virtual keyboard provides. However, loud noises during tunnel construction could make this feature unusable.

8. World locking tools also work well in a tunnel environment. However, tests with larger distances are needed to check this in more detail.

9. Spatial mesh detection works for walls in tunnels. Sometimes the process takes a while to recognize the wall. The walls are also not always fully recognized. More tests are needed with the spatial mesh detection to check how well this works. Detecting partial spatial meshes of a wall works very well.

10. Users should be able to hide windows so that the field of view is clear for their tasks. In general, hiding windows provided promising results for the users so that they could concentrate on the task at hand.

Utilizing the previous suggestions should improve the usability and the acceptance of the users for future AR applications in the environment of tunnel construction.

CHAPTER 9

Conclusion and Future Work

This chapter summarizes the thesis and the implications that arose from the results. Finally, the thesis concludes with recommendations for further research on designing and developing AR applications in tunnel construction.

9.1 Conclusion

This study investigated if localization in a tunnel using AR is viable. Furthermore, this study aimed to find problems in tunnel construction solvable with AR. Lastly, it tried to answer if the solutions developed for these problems are usable in a tunnel environment.

Regarding the localization prototype, we identified that the proposed solution delivers good results. Using the World-Locking Tools framework and space pins to map out a tunnel is a stable approach. By adding the option to use physical markers, we also provide a solution in case the environment changes so much that the solution can no longer detect the virtual markers. From the usability tests, we conclude that the prototype is intuitive enough for users to map even greater distances.

We collected problems improvable with AR applications from the focus group interview. For three of these problems, we decided to implement high-fidelity prototypes. Before the implementation, we did another interview to collect prototype requirements. During this interview, we realized that only two problems could benefit from an AR application.

One prototype we implemented is called the acceptance inspection prototype. We integrated the localization solution we implemented earlier in the acceptance inspection prototype. From the usability tests, we gathered that this solution could be beneficial for inspections. However, according to the participants, some minor adjustments are needed to be practical. These improvements include an alternative to the slow keyboard input and more guidance information for the user interface.

The second implemented prototype was called the tunnel face mapping prototype. This prototype also utilized the localization solution. The usability tests for this prototype revealed that the current implementation is not yet ready to be helpful in tunnel construction. Participants struggled with the far-interaction hand gestures to set up the tunnel face mapping. A more automated approach for this step is needed to be practical. Such automation is needed because tunnel face mappings need to happen as quickly as possible not to stall further tasks at the tunnel construction site. However, most participants found that the annotation part was intuitive.

This study revealed that the development of usable AR applications is possible using currently available hardware, i.e., the HoloLens. However, a few problems still need to be solved before AR applications are practical for tunnel construction projects. These problems include spatial mesh detection and hand gesture recognition issues, low frame rate, and a narrow field of view. However, future AR devices could solve these problems. In general, AR could improve the work in tunnel construction in the future. AR provides features such as spatial mesh detection, localization without GPS, and the possibility of having hands free to work. These features are valuable for fields such as tunneling.

9.2 Reflections on the research and findings

The following summarizes how this thesis answered the featured research questions.

1. RQ1: How practical is localization in tunnels with the currently available augmented reality hardware? We answered RQ1 in Chapter 7 through the implementation and field tests at the ZAB.

2. RQ2: What problems in tunnel construction can be improved through the adoption of augmented reality? To answer RQ2, we summarized the problems found from the focus group interview in Chapter 5.

3. RQ3: How helpful are AR solutions in tunnel construction? We answered RQ3 in Chapter 8 by summarizing the results from the usability tests conducted at the ZAB and the survey.

9.3 Future Research

Since this study used the guidelines of the design science research approach, we want to provide helpful information and guidelines for future implementations in this field of study. We propose that the mentioned localization prototype or a similar solution is tested more thoroughly for longer distances in a tunnel. A test for an extended period would also provide valuable information about how well the solution works when changes happen to the tunnel, i.e., from the construction phase to the operation and maintenance phase.

In Chapter 8, we provided guidelines to help design and research future AR applications. This guideline provides information about how to implement usable AR applications in the environment of tunnel construction. Furthermore, it also provides issues that researchers should consider when testing for usability.

List of Figures

List of Tables

Bibliography

[1] G. Kvasina, „Dokumentation bei zyklischem Tunnelvortrieb – Erhebung von wesentlichen Parametern von Bauzeit und Kosten als Grundlage für ein digitales Modell“, M.S. thesis, TU Wien, Wien, 2018.

[2] A. Mazak, R. Galler, R. Wenighofer, G. Goger, T. Bednar, C. Huemer, and M. Wimmer, „TransIT: Interdisziplinäres Forschungsprojekt zur digitalen Transformation im Tief- und Tunnelbau“, *Bauaktuell*, vol. 4, 2020.

[3] Y. Zhou, H. Luo, and Y. Yang, „Implementation of augmented reality for segment displacement inspection during tunneling construction“, *Automation in Construction*, vol. 82, 2017.

[4] A. Jakl, *Basics of ar: Anchors, keypoints & feature detection*, Aug. 2018.

[5] H. Urban, T. Irschik, C. Schranz, and C. Schönauer, „Augmented Reality im Bauwesen: Teil 2 – Baustellentaugliches Trackingsystem/Augmented Reality in Civil Engineering: Part 2 – site-compatible tracking system“, *Bauingenieur*, vol. 95, Jan. 2020.

[6] M. Huymajer, D. Operta, A. Mazak-Huemer, and C. Huemer, „The Tunneling Information Management System – A tool for documenting the tunneling process in NATM projects“, *Geomechanics and Tunnelling*, vol. 15, no. 3, 2022.

[7] A. R. Hevner, S. T. March, J. Park, and S. Ram, „Design Science in Information Systems Research“, *MIS Quarterly*, 2004.

[8] B. Kitchenham and S. Charters, „Guidelines for performing Systematic Literature Reviews in Software Engineering“, Software Engineering Group School of Computer Science and Mathematics Keele University, Tech. Rep., Jan. 2007.

[9] R. A. Krueger and M. A. Casey, „Social analysis: Selected tools and techniques“, D.C. : World Bank Group, Tech. Rep., 2002.

[10] T. Moore, K. Mckee, and P. McCoughlin, „Online focus groups and qualitative research in the social sciences: their merits and limitations in a study of housing and youth“, *People, Place and Policy*, vol. 9, Apr. 2015.

[11] H. Kallio, A.-M. Pietilä, M. Johnson, and M. Kangasniemi, „Systematic methodological review: developing a framework for a qualitative semi-structured interview guide“, *Journal of advanced nursing*, vol. 72, no. 12, 2016.

[12] C. M. Barnum, *Usability Testing Essentials*. Boston: Morgan Kaufmann, 2011.

[13] G. B. Hemphill, *Practical tunnel construction*. John Wiley & Sons, Ltd, 2012.

[14] A. Muir Wood, *Tunnelling: Management by Design*. London: CRC Press, 2000.

[15] B. Kochendörfer, J. H. Liebchen, and M. G. Viering, *Bau-Projekt-Management : Grundlagen und Vorgehensweisen*, 3rd ed. Wiesbaden: B. G. Teubner Verlag / GWV Fachverlage GmbH, Wiesbaden, 2006.

[16] W. Bergeson, S. L. Ernst, *et al.*, „Tunnel operations, maintenance, inspection, and evaluation (TOMIE) manual“, United States. Federal Highway Administration, Tech. Rep., 2015.

[17] B. Maidl, *Maschineller Tunnelbau im Schildvortrieb*, 2nd ed. Berlin: Ernst & Sohn, 2011.

[18] A. Sharafat, M. S. Khan, K. Latif, and J. Seo, „BIM-Based Tunnel Information Modeling Framework for Visualization, Management, and Simulation of Drill-and-Blast Tunneling Projects“, *Journal of Computing in Civil Engineering*, vol. 35, Dec. 2020.

[19] K. C. Zach, „Ein Datenmodell zur digitalen Dokumentation des Bauprozesses im Tunnelbau“, M.S. thesis, University of Leoben, 2021.

[20] C. Preidel, A. Borrmann, H. Exner, and M. König, „Common Data Environment“, in *Building Information Modeling: Technologische Grundlagen und industrielle Praxis*, A. Borrmann, M. König, C. Koch, and J. Beetz, Eds. Wiesbaden: Springer Fachmedien Wiesbaden, 2021.

[21] International Organization for Standardization, „DIN EN ISO 19650-1:2019-08: Organisation und Digitalisierung von Informationen zu Bauwerken und Ingenieurleistungen, einschließlich Bauwerksinformationsmodellierung (BIM)“, International Organization for Standardization, Geneva, CH, Standard, Feb. 2018.

[22] A. J. Spengler and J. Peter, *Die Methode Building Information Modeling: Schnelleinstieg Für Architekten und Bauingenieure*, ser. essentials. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2020.

[23] M. Stange, *Building Information Modelling Im Planungs- und Bauprozess: Eine Quantitative Analyse Aus Planungsökonomischer Perspektive*. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2020.

[24] M. Flora, G. Fröch, and W. Gächter, „Optimierung des Baumanagements im Untertagebau mittels digitaler Infrastruktur-Informationsmodelle“, *Bautechnik*, vol. 97, no. 11, 2020.

[25] G. Kipper and J. Rampolla, *Augmented reality: An emerging technologies guide to AR*. Waltham, Mass.: Syngress, 2012.

[26] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, „Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age“, *IEEE Transactions on Robotics*, vol. 32, no. 6, 2016.

[27] A. Jakl, *Basics of AR: SLAM – simultaneous localization and mapping*, Sep. 2018.

[28] S. Leutenegger, M. Chli, and R. Y. Siegwart, „BRISK: Binary Robust invariant scalable keypoints“, in *2011 International Conference on Computer Vision*, 2011.

[29] E. Rosten and T. Drummond, „Machine Learning for High-Speed Corner Detection“, in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[30] R. Mur-Artal and J. D. Tardós, „ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras“, *IEEE Transactions on Robotics*, vol. 33, no. 5, 2017.

[31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, „ORB: An efficient alternative to SIFT or SURF“, in *2011 International Conference on Computer Vision*, 2011.

[32] A. Naceri, D. Mazzanti, J. Bimbo, D. Prattichizzo, D. G. Caldwell, L. S. Mattos, and N. Deshpande, „Towards a Virtual Reality Interface for Remote Robotic Teleoperation“, in *2019 19th International Conference on Advanced Robotics (ICAR)*, 2019.

[33] M. Speicher, B. D. Hall, and M. Nebeling, „What is Mixed Reality?“, in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19, Glasgow, Scotland Uk: Association for Computing Machinery, 2019.

[34] L. F. de Souza Cardoso, F. C. M. Q. Mariano, and E. R. Zorzal, „A survey of industrial augmented reality“, *Computers & Industrial Engineering*, vol. 139, 2020.

[35] C. Schranz, A. Gerger, and H. Urban, „Augmented Reality im Bauwesen: Teil 1 – Anwendungs- und Anforderungsanalyse (Augmented Reality in civil engineering: Part 1 – Use-case and requirement analysis)“, *Bauingenieur*, vol. 95, Oct. 2020.

[36] K. Chen and F. Xue, „The renaissance of Augmented Reality in construction: history, present status and future directions“, *Smart and Sustainable Built Environment*, vol. ahead-of-print, Dec. 2020.

[37] J. Garbett, T. Hartley, and D. Heesom, „A multi-user collaborative BIM-AR system to support design and construction“, *Automation in Construction*, vol. 122, 2021.

[38] T. Katika, F. K. Konstantinidis, T. Papaioannou, A. Dadoukis, S. N. Bolierakis, G. Tsimiklis, and A. Amditis, „Exploiting Mixed Reality in a Next-Generation IoT ecosystem of a construction site“, in *2022 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2022.

[39] T. Samad, „Control Systems and the Internet of Things [Technical Activities]“, *IEEE Control Systems Magazine*, vol. 36, no. 1, 2016.

[40] Y.-J. Chen, Y.-S. Lai, and Y.-H. Lin, „BIM-based augmented reality inspection and maintenance of fire safety equipment“, *Automation in Construction*, vol. 110, 2020.

[41] H. Kaufmann and A. Dünser, „Summary of Usability Evaluations of an Educational Augmented Reality Application“, in *Virtual Reality*, R. Shumaker, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[42] S. Knopp, P. Klimant, and C. Allmacher, „Industrial Use Case - AR Guidance using Hololens for Assembly and Disassembly of a Modular Mold, with Live Streaming for Collaborative Support“, in *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 2019.

[43] J. Hecht, „Optical Dreams, Virtual Reality“, *Opt. Photon. News*, vol. 27, no. 6, Jun. 2016.

[44] F. Abulude, A. Akinnusotu, and A. Adeyemi, „Global Positioning System and It's Wide Applications“, *Continental J. Information Technology*, Oct. 2015.

[45] M. Unal, E. Bostanci, E. Sertalp, M. S. Guzel, and N. Kanwal, „Geo-location Based Augmented Reality Application For Cultural Heritage Using Drones“, in *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2018.

[46] A. Delić, M. Domančić, P. Vujević, N. Drljević, and I. Botički, „AuGeo: A geolocation-based augmented reality application for vocational geodesy education“, in *Proceedings ELMAR-2014*, 2014.

[47] A. Gherghina, A.-C. Olteanu, and N. Tapus, „A marker-based augmented reality system for mobile devices“, in *2013 11th RoEduNet International Conference*, 2013.

[48] S. Boonbrahm, P. Boonbrahm, and C. Kaewrat, „The Use of Marker-Based Augmented Reality in Space Measurement“, *Procedia Manufacturing*, vol. 42, 2020.

[49] J. Delmerico, H. Oleynikova, J. Nieto, and M. Pollefeys, *Enabling interaction between mixed reality and robots via cloud-based localization*, Oct. 2020.

[50] T. Varelas, A. Pentefoundas, C. Georgiadis, and D. Kehagias, „An AR Indoor Positioning System Based on Anchors“, *MATTER: International Journal of Science and Technology*, vol. 6, no. 3, 2020.

[51] S. M. Chacko and V. Kapila, „An Augmented Reality Interface for Human-Robot Interaction in Unconstrained Environments“, in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[52] K. Khoshelham, H. Tran, and D. Acharya, „Indoor Mapping Eyewear: Geometric Evaluation of Spatial Mapping Capability of Hololens“, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W13, 2019.

[53] I. Soares, R. B. Sousa, M. Petry, and A. P. Moreira, „Accuracy and Repeatability Tests on HoloLens 2 and HTC Vive“, *Multimodal Technologies and Interaction*, vol. 5, no. 8, 2021.

[54] D. Schneider, V. Biener, A. Otte, T. Gesslein, P. Gagel, C. Campos, K. Čopič Pucihar, M. Kljun, E. Ofek, M. Pahud, P. O. Kristensson, and J. Grubert, „Accuracy Evaluation of Touch Tasks in Commodity Virtual and Augmented Reality Head-Mounted Displays“, in *Symposium on Spatial User Interaction*, ser. SUI '21, Virtual Event, USA: Association for Computing Machinery, 2021.

[55] S. Tavani, A. Billi, A. Corradetti, M. Mercuri, A. Bosman, M. Cuffaro, T. Seers, and E. Carminati, „Smartphone assisted fieldwork: Towards the digital transition of geoscience fieldwork using LiDAR-equipped iPhones“, *Earth-Science Reviews*, vol. 227, 2022.

[56] T. Scargill, G. Premsankar, J. Chen, and M. Gorlatova, „Here To Stay: A Quantitative Comparison of Virtual Object Stability in Markerless Mobile AR“, in *2022 2nd International Workshop on Cyber-Physical-Human System Design and Implementation (CPHS)*, May 2022.

[57] P. Black, „Dictionary of Algorithms and Data Structures“, Oct. 1998.

[58] M. K. Kaleem, D. Verma, and M. J. Idrisi, „Generalization of Line Drawing Algorithm – An effective approach to minimize the error in the existing Bresenham's Line Drawing Algorithm“, in *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2021.

Appendix

Function	Class	Purpose
CreateQR-SpacePinAt-Position	DropSpacePin	Creates a space pin at the location of a scanned QR code.
Create-SpacePin-AtPosition	DropSpacePin	Show helper model to place a space pin.
RemoveSpace-Pin	Spacepin-Manager	Removes space pin from the scene and the space pin collection.
LoadCreated-Spacepins	SpacePin-Persistence	Load WLT settings and the saved space pins.
GetTunnel-MeterFor-OriginPoint	TunnelSpace-PinNetwork	Calculate tunnel meter for a given position.
GetNearest-Node	TunnelSpace-PinNetwork	Searches for the nearest node to the given position.
GetNeighbors-ForGiven-Position	TunnelSpace-PinNetwork	Retrieves the neighboring nodes for a given position.
GetDistance-ForOrigin-PointBetween	TunnelSpace-PinNetwork	Calculates the distance for a position between two given nodes.
GetDistance-ForOrigin-PointBetween-CutEnds	TunnelSpace-PinNetwork	Similar to GetDistanceForOriginPoint-Between but returns 0 if the distance has the same length as the whole distance.
GetClosest-PointOn-FiniteLine	TunnelSpace-PinNetwork	Calculates the position on the line between two given nodes.
CheckNode-Loop	Localization-Service	A loop that checks the user's current position every second.

StartInspection	InspectionController	Calls the InspectionReportService to start a new report.
SaveCurrentReport	InspectionReportService	Forwards the current InspectionReport to the IInspectionReportPersistence to save it.
ShowRoundInfoWindow	InspectionController	Shows the tunnel round info window.
OnCurrentRoundsRetrieved	TunnelMeterRoundManager	Event function that returns the round for the current user position.
OnUserPositionChanged	LocalizationService	Event function that returns the current user position.
GetRoundsForTunnelMeter	TunnelMeterRoundPersistence	Returns all rounds for the given tunnel chainage.
CreateTunnelRoundBorders	TunnelMeterRoundManager	Shows a border (two cuboids) to visualize the start and end of a tunnel round.
GetPositionOfTunnelMeter	TunnelSpacePinNetwork	Returns a position for a given tunnel chainage.
CreateTunnelRoundBorders	TunnelMeterRoundManager	Takes care of placing the border cuboids.
UpdateCurrentRound	ZoneDisplayController	Updates the round info window information according to the current tunnel round.
MarkPlace	InspectionController	Prepares everything so the user can start the place marker process.
OnPhotoAccepted	CameraController	Event function that provides the picture taken by the CameraController.
CameraManager_OnPhotoAccepted	InspectionController	Opens a text box for the user to add information in the mark place process.
OnNoteEditFinished	InspectionController	Event function that creates an InspectionObject to end the mark place process.
AddInspectionObjectToReport	InspectionReportService	Adds an InspectionObject to the current report.
FinishInspection	InspectionController	Shows the report preview that enables the saving process.

`Preview-Report`	`Inspection-Report-Preview-Controller`	Prepares the report preview window with the current report.
`DrawPixelsOn-Canvas`	`Canvas-Annotation-Controller`	Draws pixels on the canvas for the given position.
`Upload-Mapping`	`TunnelFace-Mapping-Manager`	Provides the taken tunnel face mapping to the `TunnelFaceMappingPersistence`.
`SaveTunnel-FaceMapping`	`Tunnel-FaceMapping-Persistence`	Saves the given mapping.

Table 1: An overview of all the functions mentioned in Chapter 7.