

Hochschule  
für Technik  
Stuttgart

University of Applied Sciences

Master of Science Programme  
Photogrammetry and Geoinformatics  
Master Thesis  
Winter Term 2021/2022

**Occlusion Screening using 3D city models as a  
reference database for mobile AR-Applications**

by

Muhammad Alfakhori

Supervisors:

Prof. Dr.-Ing. Volker Coors  
Mr. Philipp Willkomm

# **Occlusion Screening using 3D city models as a reference database for mobile AR-Applications**

by

**Muhammad Alfakhori**

**A dissertation presented in partial fulfillment of the requirements for the degree of  
Master of Science in the Department of Geomatics, Computer Science and Math-  
ematics, Stuttgart University of Applied Sciences**

## **Declaration**

The following Master thesis was prepared in my own words without any additional help. All used sources of literature are listed at the end of the thesis.

I hereby grant to Stuttgart University of Applied Sciences permission to reproduce and to distribute publicly paper and electronic copies of this document in whole and in part.

Stuttgart, 28.02.2022

\_\_\_\_\_  
Muhammad Alfakhori

Approved by:

\_\_\_\_\_  
Prof. Dr. Volker Coors

## ACKNOWLEDGEMENT

First, I would like to thank God for the grace and strength he provided me to complete this course. The project would not have been completed without the help of several people. I would like to extend my sincere gratitude and appreciation to my supervisor, Prof. Dr.-Ing. Volker Coors (HFT Stuttgart), for his persistent support and motivation. His constant encouragement and flexible working practices, combined with his constructive suggestions and guidance, always created a favorable research environment and oriented me in the right direction. My deep appreciation and gratitude are also extended to my co-supervisor, Mr. Philipp Willkomm (M.O.S.S. GmbH), for his valuable comments and suggestions throughout this thesis period.

Also, I would like to thank Habiburrahman Dastageeri, Dr. Sven Schneider, Luca Casagrande, and all the mobility4icity project team for their support.

I am grateful to all the professors of the Master Photogrammetry and Geoinformatics course, especially Prof. Dr. Dietrich Schröder, for his continuous support and keenness to have an excellent experience during our master's studies.

Thank you to all my classmates in this master's program as well for the wonderful company.

Throughout my life, I've been fortunate to have a loving and supportive family, and I'd want to take this opportunity to show my gratitude to my mother, father, brothers, and sisters for their unwavering support to my education and research, even if they didn't understand a word of it. Although we are separated geographically, your encouragement, belief in me, and moral support has always encouraged me to pursue my academic goals.

To my wonderful family, I dedicate my thesis.

## **Occlusion Screening using 3D city models as a reference database for mobile AR-Applications**

### **ABSTRACT**

Creating an immersive Augmented Reality (AR) experience requires aligning the digital content with the real environment where the digital content appears and interacts in a similar way to a real object. For that, the virtual object must persist in its position across sessions and be occluded completely or partly depending on whether a real-world object is in the line of sight. This research makes use of the cutting-edge technology available in the field of AR/MR, the Microsoft HoloLens 2.0, where the in-device Time of Flight (ToF) camera is used to scan the environment to create a Spatial Map. LOD1 city model enriches the Spatial Map and in addition, is used as an occlusion mask via a custom rendering pipeline. Spatial Anchors can be tracked using the Simultaneous localization and mapping (SLAM) used in conjunction with World Locking Tools to anchor the whole scene to the real-world coordinates. The developed occlusion effect is used in urban planning scenarios to introduce a new area design featuring a car-free environment.

The proposed method is evaluated based on performance indicators including the number of Frame Per Second (FPS) since it's highly correlated to user comfort. The findings of the users' study demonstrate that the occlusion effect achieves its purpose since most of the participants reported enhancement in the depth perception and overall experience by enabling the occlusion screening.

**Keywords:** Augmented Reality, Mixed Reality, Occlusion, HoloLens

# TABLE OF CONTENT

<b>ACKNOWLEDGEMENT .....</b>	<b>ii</b>
<b>ABSTRACT .....</b>	<b>iii</b>
<b>TABLE OF CONTENT .....</b>	<b>iv</b>
<b>TABLE OF FIGURES .....</b>	<b>vi</b>
<b>TABLE OF TABLES .....</b>	<b>viii</b>
<b>ABBREVIATIONS .....</b>	<b>ix</b>
<b>1 INTRODUCTION .....</b>	<b>1</b>
1.1 Contextual Background .....	1
1.2 Problem Statement and Motivation .....	3
1.3 Research Aim and Objectives .....	4
1.4 Thesis Organization .....	4
<b>2 LITERATURE REVIEW .....</b>	<b>6</b>
2.1 Related Works .....	6
2.2 Mobile AR .....	14
2.3 AR Development Kits .....	16
<b>3 THEORETICAL BACKGROUND .....</b>	<b>19</b>
3.1 Tracking Methods .....	19
3.2 Spatial Mapping .....	20
3.3 Coordinate Systems .....	22
3.4 Spatial Anchors .....	23
3.5 Shaders .....	24
<b>4 DATA AND RESOURCES .....</b>	<b>26</b>
4.1 Data and Data Preparation .....	26
4.2 Software .....	28
4.3 Hardware .....	28
<b>5 METHODOLOGY .....</b>	<b>31</b>
5.1 Proposed Methodology .....	31
5.2 Evaluation Criteria .....	33
<b>6 IMPLEMENTATION .....</b>	<b>35</b>
6.1 Project Setup .....	35
6.2 Spatial Observer .....	35

6.3	Anchoring .....	37
6.4	Occlusion Shader .....	39
6.5	Designing the 3D Content.....	40
6.6	Compiling and Deploying the Application.....	42
<b>7</b>	<b>RESULTS AND EVALUATION .....</b>	<b>43</b>
7.1	Tracking System and Anchoring .....	43
7.2	Occlusion Screening .....	45
7.3	Performance Evaluation.....	48
7.4	Users' Study Feedback.....	49
<b>8</b>	<b>RESULTS DISCUSSION.....</b>	<b>52</b>
<b>9</b>	<b>CONCLUSION AND FUTURE WORKS .....</b>	<b>55</b>
	<b>REFERENCES.....</b>	<b>57</b>
	<b>ANNEXES.....</b>	<b>64</b>
I.	Fine-Tuning Alignment Dashboard Script.....	64
II.	Occlusion Shader Script.....	66
III.	User Menu Script.....	67

## TABLE OF FIGURES

Figure 1-1: Reality-virtuality continuum. ....	1
Figure 1-2: Demonstration of the occlusion effect. ....	2
Figure 1-3: Bird's eye view of the study area. ....	3
Figure 2-1: Allen et al. AR-Application GUI. ....	7
Figure 2-2: Grayscale filter visualization for outdoor AR application. ....	8
Figure 2-3: 3D building model visualization using novaFACTORY and HoloLens. ....	8
Figure 2-4: Geospatial data visualization using novaFACTORY and HoloLens. ....	9
Figure 2-5: Occlusion handling using a model-based approach. ....	10
Figure 2-6: Occlusion handling based on OSM data. ....	11
Figure 2-7: Using 3D city model for outdoor AR tracking system. ....	12
Figure 2-8: Using HoloLens in operation rooms. ....	12
Figure 2-9: Real-time holographic dashboard showing the temperature of a pipe. ....	13
Figure 2-10: Recognized objects with their corresponding labels. ....	14
Figure 2-11: Different AR glasses. ....	16
Figure 3-1: SLAM algorithm overview. ....	21
Figure 3-2: Cartesian coordinate systems. ....	22
Figure 3-3: Relationship between Spatial Anchor and keypoints. ....	23
Figure 3-4: The relationship between 3D models, shader, and materials in Unity3D. ....	24
Figure 4-1: CityGML 2.0 building model classification based on LOD. ....	26
Figure 4-2: FME pipeline to convert CityGML to OBJ mesh. ....	27
Figure 4-3: 3D mesh building model of the study area. ....	27
Figure 4-4: HoloLens 2.0 camera configuration. ....	29
Figure 5-1: An overview of the workflow of the proposed method. ....	31
Figure 5-2: World Locking Tools architecture. ....	32
Figure 6-1: Spatial map of an indoor environment captured by the HoloLens. ....	36
Figure 6-2: Spatial map mesh superimposed over the real environment scene. ....	36
Figure 6-3: World Locking Tools. ....	37
Figure 6-4: World Locking Tools in-application dashboard. ....	38
Figure 6-5: Alignment fine-tuning dashboard. ....	38
Figure 6-6: User menu to control occlusion screening behavior. ....	40
Figure 6-7: Proposed concept draft to redesign Züblin parking garage area. ....	40
Figure 6-8: 3D model of a proposed building to replace the Züblin parking garage. ....	41
Figure 6-9: The 3D-designed grass hedge and bushes. ....	42
Figure 7-1: Localizing a test object using Vuforia Area Target. ....	43
Figure 7-2: Distributed Spatial Anchors in the study area. ....	44
Figure 7-3: Study area before and after superimposing the augmented content. ....	45
Figure 7-4: Occlusion mask based on scanned spatial map and 3D building model. ....	45
Figure 7-5: Occlusion screening based on the spatial map. ....	46
Figure 7-6: Noises during spatial map scanning caused by moving pedestrians and passing cars. ....	46
Figure 7-7: Occlusion screening based on the 3D building model. ....	47
Figure 7-8: Alignment of the 3D building model occlusion mask with real environment. ....	47

Figure 7-9: Comparison between occlusion screening based on the spatial map and the 3D building model. ....	48
Figure 7-10: Relation between the number of triangles and rendered FPS.....	49
Figure 7-11: Participants' answers to the question about their prior knowledge.....	50
Figure 7-12: Summary of the users' study feedback. ....	51



## TABLE OF TABLES

Table 2-1: Comparison of common AR SDKs. ....	18
Table 4-1: HoloLens camera characteristics. ....	29
Table 7-1: Number of Spatial Anchors, linear and angular deviation during test sessions.....	44

## ABBREVIATIONS

AR	Augmented Reality
DoF	Degrees of Freedom
FOV	Field of View
FPS	Frame Per Second
GUI	Graphical User Interface
HMD	Head-Mounted Display
IMU	Inertial Measurement Unit
LOD	Level of Detail
MR	Mixed Reality
MRTK	Mixed Reality Toolkit
ORB	Oriented FAST Rotated BRIEF
OSM	Open Street Map
SDK	Software Development Kit
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded-Up Robust Features
ToF	Time of Flight
UWP	Universal Windows Platform
VR	Virtual Reality

# 1 INTRODUCTION

## 1.1 Contextual Background

Augmented Reality (AR) refers to the science and technology to overlay digital content over the real environment. In other words, AR allows users to view material that they would otherwise be unable to see [1]. AR can be extended further to have a more interactive and immersive experience by introducing an interaction between digital content and the real-world environment to achieve a Mixed Reality (MR) and near-real experience. This necessitates a seamless transition between virtual and physical environments, as well as the ability for both to coexist and interact realistically. Simulating physical interactions like collisions, shadows, lighting, and occlusions is a difficult issue to overcome [1], [2]. For that, the digital content should align with the real environment including position, scale, and occlusion by other objects.

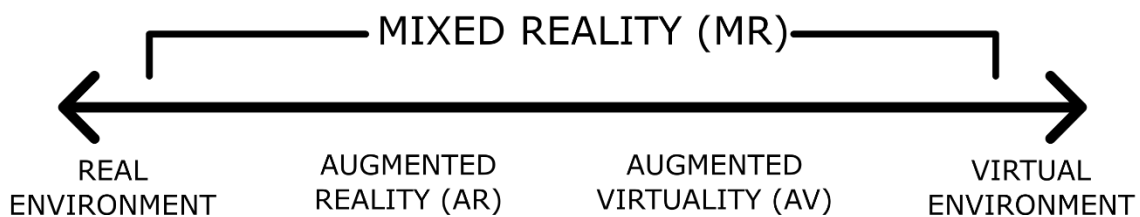


Figure 1-1: Reality-virtuality continuum.

The relationship between the real and virtual environments is shown in Figure 1-1, with the AR environment situated anywhere in between depending on the presence of the digital content. This indicates that as we draw closer to virtual reality, we are moving away from the real environment, and vice versa.

When building AR scenes, the purpose of occlusion is to maintain the laws of the line of sight. That is, any virtual item that is placed behind a real object should be occluded or hidden behind the actual object to provide the viewer with a realistic experience and improve the viewer's depth perception. As shown in Figure 1-2, when no occlusion effect is applied, the digital content seems to be floating above the scene; however, when the occlusion effect is applied and part of the objects is hidden, the scene appears more realistic, and the digital content blends with the real environment.



Figure 1-2: Demonstration of the occlusion effect<sup>1</sup>. (*left*) Disabled. (*right*) Enabled.

To achieve occlusion, knowledge of the real environment is required through the presence of a three-dimensional representation of it, which in turn will act as a mask to hide the virtual objects. This real-world representation can be obtained in real-time using various 3D sensing techniques or can be obtained by using a generalized model in the form of a city model, which can be obtained in advance.

With the rapid growth of AR technologies and their numerous use cases including architecture and urban planning, it is essential that novel inventive methods be introduced, as well as improvements in performance metrics and their simple accessibility. Since urban planning plays an important role in our lives, new technologies such as AR have the potential to make significant contributions to this field of study. Increasing the use of qualitative AR solutions is essential to break free from their current limited range of applications, which are primarily focused on two-dimensional data visualization or marker-based systems [3].

Participation in this research is part of the iCity 2: Mobility4iCity project, which aims to contribute to promoting sustainable mobility, qualification of urban space, and protection against low-frequency noise in the Stuttgart region, as well as developing preliminary concepts for each of these goals. For the purposes of this study, the neighborhoods of Bohnenviertel and Leonhardsviertel shown in Figure 1-3 are used as study cases, as they both have high urban density and a scarcity of outdoor green space. Where the AR application is used to show local residents

---

<sup>1</sup> Image source: <https://developers.google.com/ar/develop/java/depth/introduction>

how the new design of the area will look and get their feedback, which can then be used in the decision-making process at a later stage.

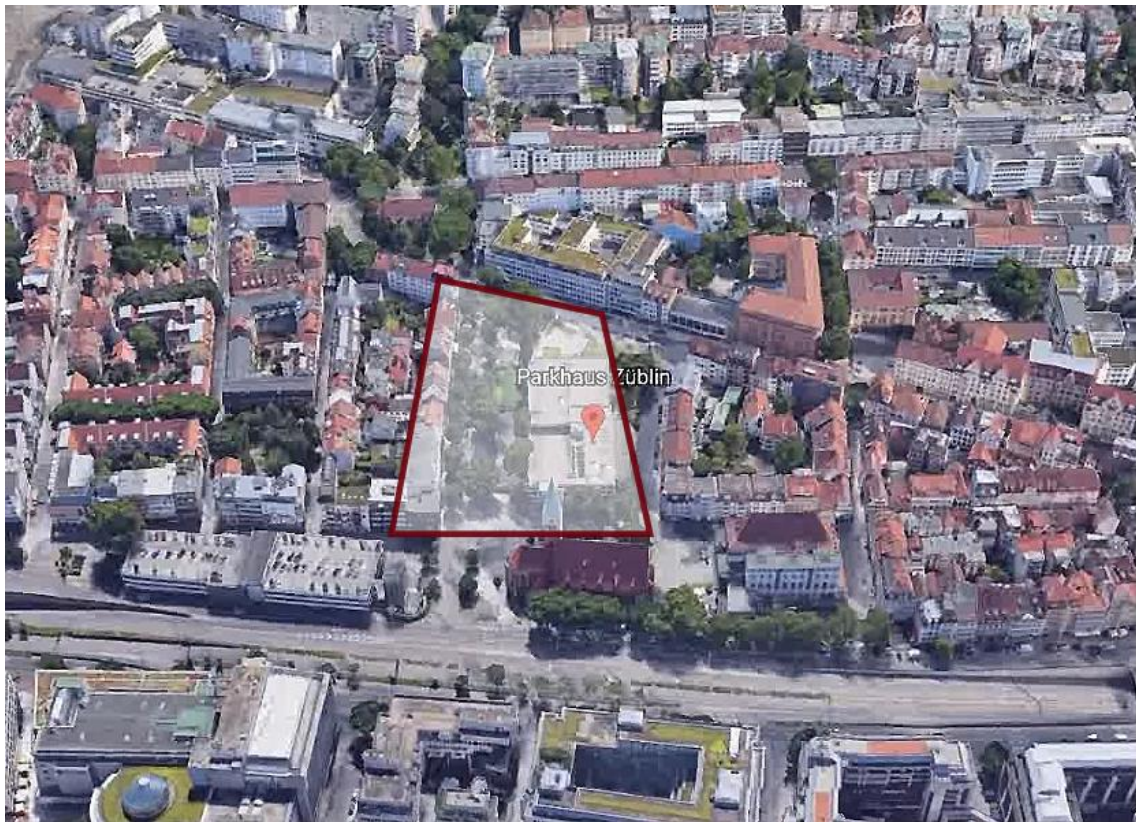


Figure 1-3: Bird's eye view of the study area<sup>2</sup>.

## 1.2 Problem Statement and Motivation

The majority of occlusion screening research is focused on achieving pixel-perfect results, for this, studies either use real-time 3D sensing technology or a pre-existing model of the environment. Neither of the research was able to effectively integrate the two methodologies in its findings. Additionally, a prior study focused on the use of sensor-based tracking systems, which were found to be insufficiently accurate [1], which might have an impact on the entire AR experience.

This research intends to make use of the most up-to-date state-of-the-art technology available in the field of AR/MR represented by the Microsoft HoloLens 2.0. Since the device does not have a GPS tracking system, it can only handle local coordinates, which implies that a computer-vision approach must be used to anchor the digital content to its intended location.

---

<sup>2</sup> Image source: Google Earth



Furthermore, the HoloLens 2.0 uses a Time of Flight (ToF) camera to scan the environment, which may be utilized to create an occlusion screening effect in some cases. The range of the ToF camera is restricted to five meters, which implies that although it performs well in indoor scenarios, it is unsuitable for outdoor application, particularly in a scenario such as urban planning. The Spatial Map, which could be scanned using the in-device ToF camera, would need to be further enhanced with a 3D city model to have a sufficient occlusion effect.

### **1.3 Research Aim and Objectives**

In response to creating a near-real-world and immersive experience for the user, the main objective of this research is to address the occlusion issue, which occurs when virtual content is hidden behind a real object, by enriching the in-device scanned Spatial Map with a 3D building model. In addition, place digital content in the intended location and maintain the position across multiple application sessions to create an AR experience that reflects a car-free environment and can be incorporated into the urban planning process.

This research aims to answer the following questions:

- How to anchor digital content to the real-world coordinates system in the study area?
- How to use in-device scanning capability for dynamic object occlusion?
- Is it possible to enrich the scanned and reconstructed spatial mesh map with a 3D building model to improve the outdoor occlusion effect?

### **1.4 Thesis Organization**

This thesis is divided into nine chapters, each of which describes the many tasks that were undertaken to complete it. Chapter 1 entails the foundation of the thesis starting with the first section briefing about the contextual background on the AR and occlusion followed by introducing the study area and the use case. The second section discusses the gaps in previous research on occlusion screening, followed by the challenges associated with this thesis when working in an outdoor environment. The third section states the main aim of this thesis along with the specific objectives set to achieve it.

Chapter 2 in the first section provides an in-depth examination of the state-of-the-art in AR technology, based on current research and development projects. The second section discusses the advancement of mobile AR and assesses the solutions that are currently available. The last section goes through some of the most popular AR Software Development Kits (SDKs) and compares their characteristics.

Chapter 3 introduces a theoretical background and explains the foundations of various tracking systems, as well as gives an overview of simultaneous localization and mapping. Followed by a discussion of 3D software and game engine coordinates systems. The third section explains in depth the concept of Spatial Anchors. Finally, the last section highlights the concept of shaders and their importance.

Chapter 4 begins with a description of the data source, followed by a detailed discussion of how the data was prepared for this research. The second section provides an overview of the software that was used, including the AR SDKs and code compilation tools. The last section discusses the platform used and its key characteristics.

Chapter 5 describes the proposed method for this thesis work and overall workflow. Afterward discusses the evaluation criteria used to evaluate the proposed method in terms of performance and users' feedback.

Chapter 6 is concerned with the implementation aspects of the methodology, including a description of the project setup and an explanation of the custom shader that was used, as well as the design of the 3D content itself.

Chapter 7 presents the results and evaluation of the proposed methodology. The first section consists of the output of the tracking system and anchoring methods. The second section illustrates the occlusion effect based on the in-device spatial map and the 3D building model and provides a visual comparison of both. The third section evaluates the performance of the developed application. The fourth section reports the feedback gathered from the users' study.

Chapter 8 discusses the significance of the results. It further presents the comparison of this thesis with the similar works done before and presents the differences between these methods and the distinction of the proposed method with them.

In the last chapter, a conclusion is drawn based on the implementation process and the results obtained. In addition, the limitations of the study and recommendations for possible future work are highlighted.

The content afterward consists of Annexes to supplement the contents described shortly in the main chapters for the page limitation.

## **2 LITERATURE REVIEW**

This chapter provides an in-depth examination of the current state of the art in AR technology, which is based on current research and development projects. First, the relevance of AR in architecture and urban planning and its potential use in history teaching and tourism, are discussed. Following that, an assessment of the occlusion handling and the use of 3D city models will be conducted. Alongside that, there will be an investigation into the many applications of Microsoft HoloLens in medical and industrial digital twin usage. The second section discusses the advancement of mobile AR and assesses the solutions that are currently available. The third and last section goes through some of the most popular AR SDKs and compares their characteristics.

### **2.1 Related Works**

#### **2.1.1 AR for Architecture and Urban Planning**

To increase the degree of immersion in the urban planning solution and to enable urban planning professionals to travel about city streets while projecting virtual 3D buildings, enabling them to view both the actual city and the virtual structures concurrently, different AR applications have been implemented. Allen et al. [4] have shown that such interactive solutions are an excellent approach to improve public engagement in urban planning processes. Allen et al. developed a smartphone application to visualize the proposed architecture design superimposed on the existing real-world environment. The users were able to view the proposed 3D architectural model and give their feedback based on their personal preferences. As such, the Graphical User Interface (GUI) as shown in Figure 2-1 was needed to include these features in the most straightforward and user-friendly manner feasible. The researchers found that familiarity with smartphones affected the users' experience directly.





Figure 2-1: Allen et al. AR-Application GUI [4].

The City 3D-AR pilot project aims to place 3D objects in the existing environment by using GPS longitude and latitude information. The project's primary problems include representing large 3D objects in an outdoor environment and detecting the location of buildings based on the participant's distance and viewing angle. The conventional technique based on fiducial markers is inapplicable; it is only suited for indoor solutions over short distances [3]. For that, a set consisting of a notebook for processing and rendering connected to a USB GPS sensor was used. A head-mounted display with the functionality of augmented reality (Vuzix Wrap 920AR) was used to visualize the output. The viewer location is calculated continuously to support viewer movement and automated 3D object transformations, rotations, and scaling in response to a view's changed angle and distance. To be able to achieve an interactive AR experience, a database of different 3D buildings is added to provide the viewer with the ability to change different architectural models. Also, the application supports the selection of different colors and materials for each building.

After major earthquakes in Christchurch, New Zealand in 2011, CityViewAR was developed to visualize panorama photographs of the city in the aftermath of the earthquakes [5]. The application was created to help in the recovery of destroyed structures and the city's future development. A GPS-enabled smartphone is needed to guide the viewer to different locations in the panoramas. The panoramic images rotate as the viewer changes their view angle based on the in-device gyroscope sensor.

Hui in his master's thesis [6] proposed an outdoor mobile AR solution for focus visualization using a grayscale filter shown in Figure 2-2. Urban planners frequently employ a grayscale

stencil filter to emphasize a particular urban element without overwhelming information. His method uses image-based matching and real-time masking using ARCore SDK.



Figure 2-2: Grayscale filter visualization for outdoor AR application [6].

As a more commercially oriented focus, novaFACTORY is an application developed by M.O.S.S. Computer Grafik Systeme GmbH enables visualization of geospatial data, including orthophotos, topographic maps, terrain models, landscape models, and 3D buildings. novaFACTORY uses a custom JSON format where different data types can be converted using a server-based solution. Figure 2-3 shows how 3D building model data can be visualized on top of orthophoto using novaFACTORY.

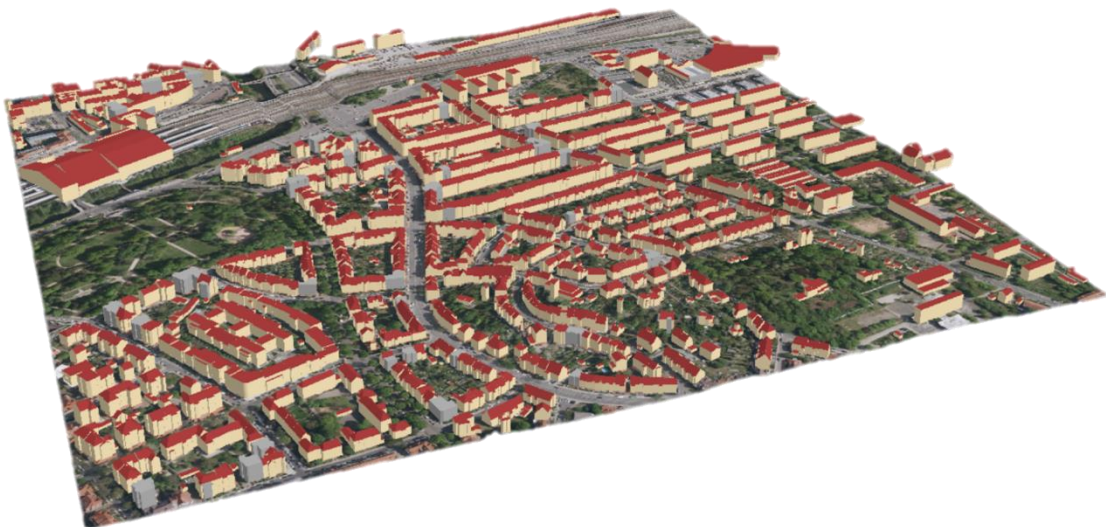


Figure 2-3: 3D building model visualization using novaFACTORY and HoloLens.



Figure 2-4: Geospatial data visualization using novaFACTORY and HoloLens.

novaFACTORY enables the users to interact with the 3D content by moving, rotating, and scaling using hand gestures or an XBOX controller as shown in Figure 2-4.

### **2.1.2 AR as a Medium for Teaching History and Tourism**

The conventional method of teaching history has been enhanced by the inclusion of a range of historical videos, photographs, and interviews with current witnesses. Mobile AR technology, on the other hand, provides far more intriguing information in one location in real-time [7]. A variety of multimedia can be placed on top of the surrounding environment, including but not limited to images, videos, audio, text, and annotations.

Intelligent Tourism and Cultural Information through Ubiquitous Services (iTACITUS), an AR-based project created by the European Commission, intends to deliver an individualized cultural travel experience for each person, based on a distributed storehouse of cultural (e.g., historical, scientific, and archaeological) information [8].

In the Czech Republic, a pilot project called “Memory of Nations” includes approximately 200 locations, each with a detailed description of the site and incident from eyewitnesses [9]. Users of the application may listen to remarks and read about significant events that occurred in the specified area.

AR has been implemented in the SPIRIT project to design a concept for imparting knowledge to historical sites [10]. The technical part consists of the development of an interactive mobile location-based augmented reality end application based on modern tablet PCs or smartphones.

By using a camera, GPS, and other sensors, historical material is visualized in a location-specific way that is controlled by a narrative engine and presented in multimedia.

The Petersen Automotive Museum in Los Angeles created an AR experience based on Microsoft HoloLens. The main objective was to highlight notable vehicles in their collection [11]. “Hollywood Dream Machines: Vehicles of Science Fiction and Fantasy” showcases some of the most recognized automobiles of all time.

### 2.1.3 Occlusion Handling in AR

Occlusion is a highly effective 3D cue. It is, in fact, the most fundamental depth indicator [12]. Therefore, to be able to achieve a seamless transition between virtual content and the real-world environment, physical simulation is needed [2].

Fortin and Hebert [2] presented two approaches for real-time occlusion handling. The first approach is model-based, where wrapping actual items in an approximate bounding box are the concept. For that reason, the tracking camera shown in Figure 2-5 is used to locate the real object target. After that, a segmentation mask is generated by subtracting the background. The first approach was more suitable for a static viewpoint.

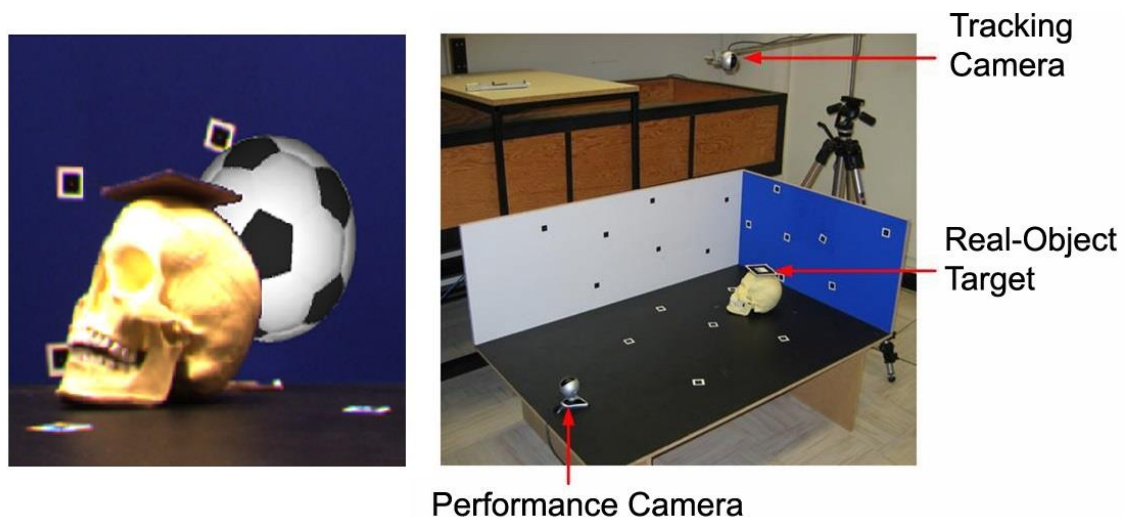


Figure 2-5: Occlusion handling using a model-based approach. (*left*) a virtual object placed behind the occluding object. (*right*) Prototype of the tracking and performance cameras [2].

In the second approach, the performance camera is replaced by a pair of stereo cameras to generate a depth map. Then, the depth is used for occlusion masking regarding the viewpoint. Additionally, the depth-based technique eliminates the need for a predefined model and enables the ability to occlude unknown objects.



Kasper et al. introduce a new method [1] for occlusion masking using Open Street Map (OSM) data for outdoor usage as shown in Figure 2-6. Kasper stressed the need for 3D representation of the real-world environment. The model can be created manually, defined by the user, or reconstructed by using depth sensors. In his method, Kasper removes the target regions from the virtual building using masks constructed from OSM buildings' data. Due to missing height information in OSM building data and inaccuracy in the position due to GPS sensor errors, the obtained results were not pixel-perfect and full occlusion was not achieved.

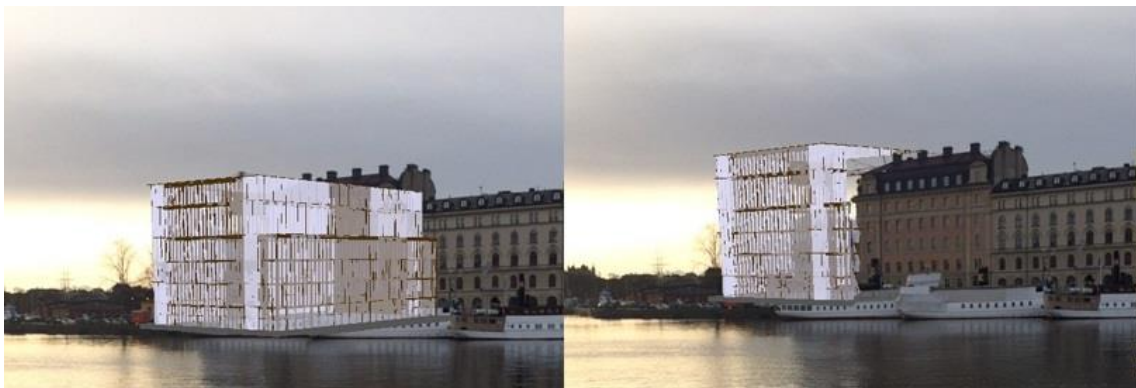


Figure 2-6: Occlusion handling based on OSM data. (*left*) Occlusion deactivated. (*right*) Occlusion activated [1].

More depth-sensor-oriented research highlighted the use of ToF cameras and LiDAR technology. Fisher et al. [13] used a ToF camera to generate a 2D depth map, which is used as an occlusion handler. However, Fisher's results highlighted some potential flaws since the data retrieved from the ToF camera often contains noise. Behzadan and Kamat [14] use LiDAR technology to overcome incorrect occlusion handling.

#### **2.1.4 Using of 3D City Model**

The development of computational capabilities and the increase in the number of 3D city models facilitated their integration into AR technology. Le and Fan [15] provided an AR tracking testbed based on a 3D city model. Their research focused on overcoming the GPS sensor drift over a short period [15], especially in built-up areas. For that, a computer vision approach was followed, where Speeded-Up Robust Features (SURF) were used to detect distinct features. A rotation matrix can be calculated based on the coordinated 3D feature points extracted from the 3D city model and the 2D feature points from the live view video. Then, the virtual content can be overlaid onto the real-world environment. Figure 2-7 shows the registration of virtual objects in the live video frame, where red circles represent matched SURF feature points between the

video frame and the associated 3D points from the produced 3D city model, and the blue sign represents the virtual items.



Figure 2-7: Using 3D city model for outdoor AR tracking system [15].

### 2.1.5 Microsoft HoloLens applications

In the first quarter of 2016, Microsoft unveiled HoloLens, the first commercially accessible MR headset, followed by version 2.0 in late 2019. An in-depth description of it will be presented later in section 4.3. Since then, different applications have been developed using the HoloLens as a platform. One of them is the Petersen Automotive Museum, mentioned in section 2.1.2.

Tepper et al. [16] discuss how wearable technologies such as HoloLens enable surgeons to access multimodal information in real-time without interfering with their workflow or surgical efficiency, as shown in Figure 2-8.

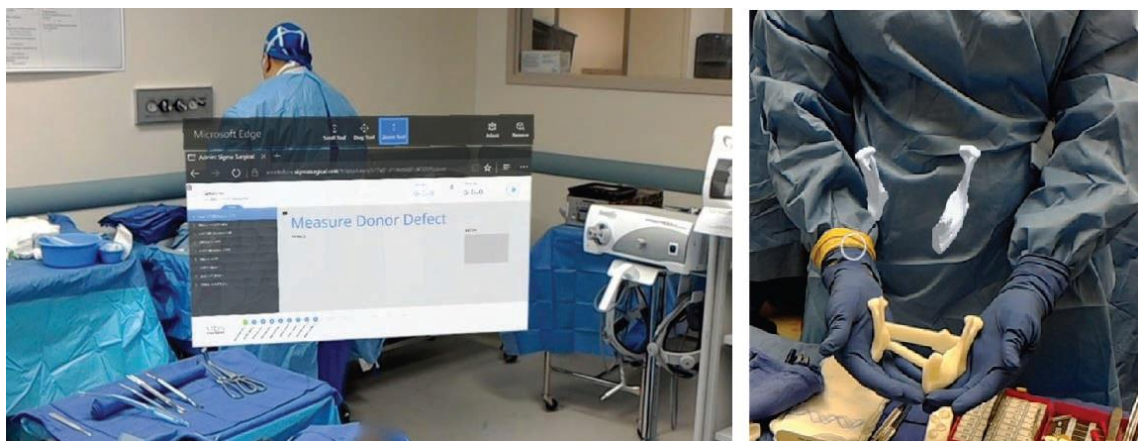


Figure 2-8: Using HoloLens in operation rooms. (left) Surgeons use HoloLens to display operation procedures. (right) 3D model of the mandible [16].

Since the early 1990s, industrial applications of augmented reality have attracted significant scientific interest [17]. Jakl et al. developed an architecture to embed holographic dashboards in the physical environment to see real-time machine data obtained through industry-standard protocols from its digital twin [17]. Figure 2-9 shows instantaneous temperature data for a pipe visualized using HoloLens.



Figure 2-9: Real-time holographic dashboard showing the temperature of a pipe [17].

Using two case studies, Filipenk et al. [18] demonstrate how the fields of industrial robots and MR may work together. The purpose of those case studies is to get a better understanding of how magnetic resonance may be beneficial and what its limits are. The first case study outlines a method for seeing a robot arm's digital twin. On the other hand, the second case study tries to make the operation of industrial robots easier by using a virtual assistant. During their studies, they found that the processing power of the HoloLens is not sufficient to render complex 3D content. Another issue was the alignment of virtual content, which Liu mentioned in his technical evaluation of HoloLens [19].

Mercedes-Benz, a German automaker, announced [20] in September 2020 their intention to equip authorized American workshops with HoloLens to speed up vehicle maintenance by providing a remote assistant.

By integrating Convolutional Neural Networks with the HoloLens, Farasin et al. were able to develop a real-time object detection application. They used the Oriented FAST Rotated BRIEF (ORB) algorithm to detect changes between two frames. When the application detects changes in the view (e.g., a new object appears), the frame captured by the HoloLens webcam will be sent to a cloud-based object detection service. After that, the label of the recognized object will be displayed over it [21]. Figure 2-10 shows recognized objects in two different scenes where each label is placed over the respective object.

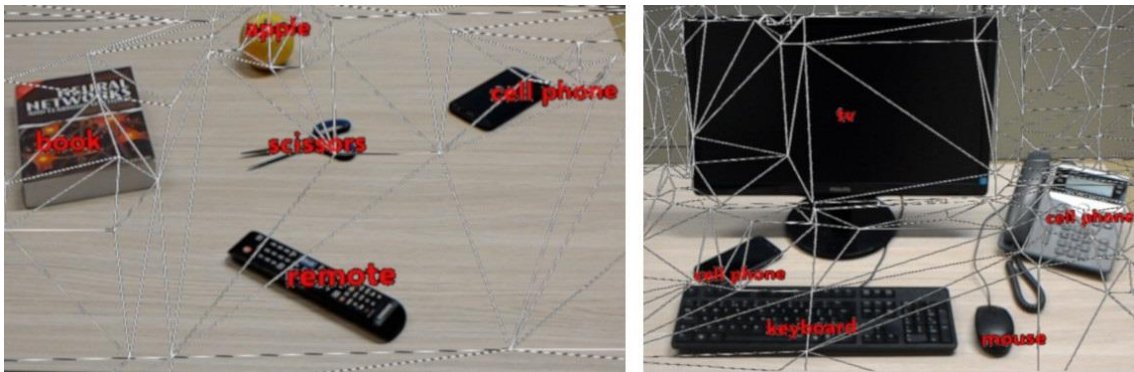


Figure 2-10: Recognized objects with their corresponding labels [21].

## 2.2 Mobile AR

Earlier AR systems need costly hardware and bespoke programming [1], and while the term augmented reality was coined in the early 1990s [22], the first fully functional AR system dates back to the late 1960s [23] when Sutherland et al. developed a mechanically tracked three-dimensional see-through headset that allowed the wearer to view computer-generated information mixed with physical objects [24].

Entertainment companies stepped in with blockbusters like the Terminator series, which depicted computer-generated graphic content. In the 1970s and 1980s, AR was studied at the US Air Force's Armstrong Laboratory, NASA's Ames Research Center, MIT, and UNC-Chapel Hill. Furness created a high-resolution heads-up overlay display with 3D sound for the US Air Force Super Cockpit project [25].

To have a higher mobility system for outdoor applications, Coors and Jasnoch developed a compact wearable GIS solution [26]. At any moment, the proposed solution could provide information on historical structures and other points of interest.

However, as computers get more powerful and compact, new mobile, wearable, and ubiquitous computing applications become more feasible [22]. The development of smartphones and the increase in the number of users create new opportunities for developers to create AR applications. According to Statista, more than 6.6 billion people used smartphones in 2022 and that number is projected to increase by another billion by 2026 [27].

Various companies have recently presented their products in conjunction with the development of Head-Mounted Display (HMD). Oculus launched the Rift, a PC-powered VR headset with 6 Degrees of Freedom (DoF) to track the movement of both your head and body. Followed by



Quest, an all-in-one VR headset eliminates the need for a computer. The latest software update of the Quest 2 brings video-based AR capability.

Other corporations like HTC, Valve, and Sony Corporations have provided VR headsets for gaming purposes. While VR technologies show promise, fundamental restrictions have hampered their general acceptance. One such barrier is the exclusion of the real-world environment [16]. Google announced an all-in-one AR see-through headset called Google Glass with limited one-hand control, but later, Glass was discontinued and is not publicly available. In 2016, Microsoft released HoloLens, a self-contained, wearable, AR glass that allows the user to display interactive 3D holograms in the immediate vicinity with the support of a natural user interface [28].

Vuzix also offers an AR wearable solution, which includes the M300 and M400. Their primary focus is on industrial applications, where the device can be mounted on top of ordinary or safety glasses. The device is equipped with a GPS sensor as well as head tracking functionality. Android serves as the operating system for the device, and it has gesture-based touch controls in addition to a touchpad. RealWear also offers a solution that is identical to this one and has the same specifications.

Magic Leap is another firm that is putting forward attempts to develop AR. They created its first headgear, the Magic Leap One glasses, which can overlay digital 3D content on top of the real-world environment in a similar way to how Google Glass and Microsoft HoloLens do their thing.

One option that is more oriented towards the end customer is offered by Facebook in cooperation with Ray-Ban, who have developed a standard sunglass that has limited augmented reality capabilities. A smartphone with the Facebook application installed is required to use the augmented reality features. The device itself is equipped with a camera for taking photos and videos, as well as a speaker for listening to various Facebook material.

Figure 2-11 shows the different AR glasses mentioned above.

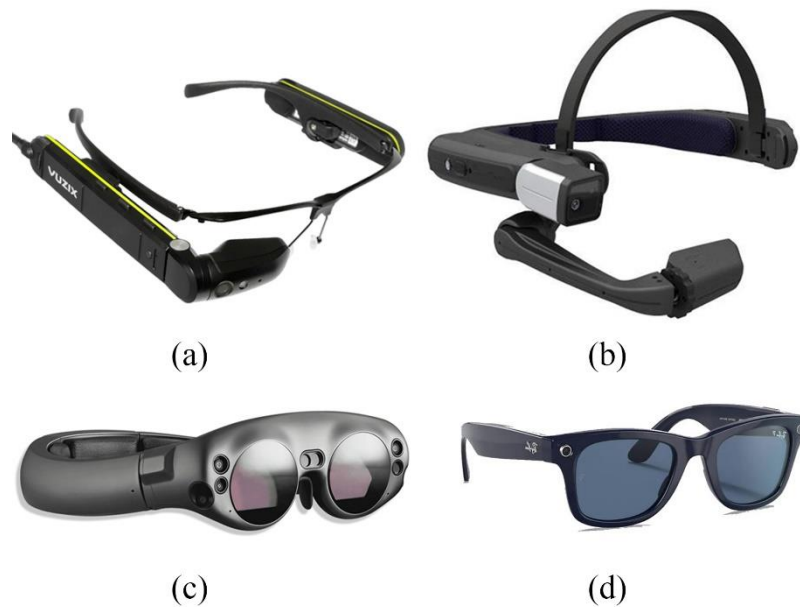


Figure 2-11: Different AR glasses. (a) Vuzix M300<sup>3</sup> (b) RealWear HMT-1<sup>4</sup> (c) Magic Leap 1<sup>5</sup> (d) Ray-Ban Stories Smart Glasses<sup>6</sup>.

## 2.3 AR Development Kits

Technological development in hardware has boosted the development of different Software Development Kits (SDKs), as developers can use different libraries depending on the application and the platform used. SDKs for augmented reality may be classified into various major categories: Sensor-based SDKs use GPS and Inertial Measurement Unit (IMU). Marker-based SDKs rely on special design images, e.g., QR codes, to generate the AR experience. The latest state-of-the-art methods use natural feature tracking SDKs to detect and track distinct features in the environment [1] [29].

Amin and Govilkar [29] and Dias et al. [30] gave a detailed evaluation of several augmented reality software development kits (SDKs), including the features and limitations of each one.

ARToolKit, an open-source cross-platform AR library, was one of the first AR SDKs released in 1999 [31]. The library is written in C and uses OpenGL for rendering as well as support for marker and plain image tracking using natural feature markers. Since ARToolKit is open-

---

<sup>3</sup> Image source: <https://www.vuzix.com/products/m300-smart-glasses>

<sup>4</sup> Image source: <https://www.realwear.com>

<sup>5</sup> Image source: <https://www.magicleap.com/en-us/magic-leap-1>

<sup>6</sup> Image source: <https://www.ray-ban.com/usa/ray-ban-stories>

source, different forks were developed based on it and provided additional features. One of them is ARToolKit+, which was recorded using C++ for easier use [31].

In 2017, Apple and Google, two mobile industry titans, introduced two competitive SDKs for developing AR applications for mobile devices: ARKit and ARCore. This SDK enables users of iOS and Android devices to create immersive applications [32]. ARKit is only available on iOS devices where it can make use of the camera and other sensors like LiDAR data to understand the surrounding environment and place AR content according to that [33].

On the other hand, ARCore supports Android and iOS devices as well. Using the depth-from-motion algorithm and the Depth API, ARCore can create a 3D representation of the scene, which enables a seamless blend of the virtual content with the real-world environment [34].

Nowacki and Woda [32] compare the capabilities of both SDKs, where ARCore achieved higher plane detection accuracy and faster mapping for large surfaces. Nevertheless, ARKit was faster at detecting the first plane. The research [32] also pointed out different hardware configuration plays a role when it comes to CPU and memory usage.

Other SDKs, such as Vuforia, are focused on the real-time recognition and tracking of 2D and 3D targets. WebAR and Argon.js focus on providing an AR experience through web technologies like WebGL, HTML, CSS, and JavaScript [1] regardless of the used platform. Mixed Reality Toolkit (MRTK) is a collection of components and scripts meant to speed up the development of AR applications for the Microsoft HoloLens and other Windows Mixed Reality headsets. Both Vuforia and MRTK will be discussed in-depth in section 4.2. Table 2-1 summarizes the features and differences between common AR SDKs.

Table 2-1: Comparison of common AR SDKs [6], [29], [30], [32].

SDK	ARToolKit	ARCore	ARKit	Vuforia	MRTK
License	Open source/Free	Open source/Free	Closed Source/Free	Closed Source/Free/Commercial	Closed Source/Free
Platform	Cross-platform	Android/iOS	iOS	Cross-platform	UWP
Sensors Support	-	GPS/IMU	GPS/IMU/LiDAR	-	IMU
2D Tracking	Image Based	Image/Marker Based	Image/Marker Based	Image/Marker Based	Marker Based
3D Tracking	-	Plane Detection/Depth API	Plane and Model Detection/LiDAR	Plane and Model Detection/Area Target	SLAM

## 3 THEORETICAL BACKGROUND

This chapter introduces a theoretical background and explains the foundations of various tracking systems, as well as simultaneous localization and mapping. Followed by a discussion of 3D software and game engine coordinates systems. The third section explains in depth the concept of Spatial Anchors. Finally, the last section highlights the concept of shaders and their importance.

### 3.1 Tracking Methods

To be able to visualize the AR content according to their location concerning the real world, the AR device position and rotation should be known [1], [5]. Tracking technologies may be split into three types: sensor-based, computer-vision-based, and hybrid. Sensor-based tracking involves the usage of GPS sensors, magnetometers, and gyroscopes to determine the camera position and Field of View (FOV). Although the great mobility of this solution makes it suited for outdoor AR, research has shown that GPS accuracy limits the user experience and results in incorrect visualization [1], [3], [15].

The computer-vision-based tracking, on the other hand, makes use of the camera frames to estimate its location and rotation. Further subdividing may be done depending on the necessity for special markers placed in the environment; however, the downside of this is that they must be always visible and cannot be obscured by other objects in the environment [29].

Using Natural Feature extraction eliminates the need for markers and relies on the interest points and their descriptors for detection. Keypoints in computer vision refer to points that exhibit a significant change in one or more image properties at the same time (e.g., intensity, color, texture) such as edges, corners [35]. Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) are both influential floating algorithms for interest point detection. Even though the SURF algorithm is faster in detecting interest points [15], [35] but still both are considered slow for real-time AR applications [35], [36].

The most crucial aspect of a good feature point for an AR application is its robustness. To be effective, an algorithm must be able to locate the same interest point under various viewing situations. When you're utilizing an AR app, a lot may change including camera angle, rotation, size, lightning [36].

Detecting features is a multi-step procedure. Its constituents vary according to the algorithms used. A concise explanation of a typical detection method is as follows: First keypoint detection

where it is typical to blur the frame image in order to make the feature point candidates scale-invariant and less sensitive to noise. Scale independence is improved by using several scaled copies of the image. Then, the keypoint descriptor is defined to find the same feature again in a different image [36].

Regrettably, blurring each frame is computationally intensive in real-time augmented reality systems. For that, ARCore, ARKit, and MRTK AR SDKs mentioned in section 2.3 are using more efficient proprietary algorithms for keypoints detection [29], [36].

The hybrid-based tracking approach is a mixture of the two preceding tracking methods [1]. Besides GPS, gyroscope, and accelerometer data, this technique also includes data from additional sensors. With all of the information, we can compute what should be augmented in the FOV without having to do any actual processing on the image feed, but the actual image is still utilized for the layer of augmentation placement, which is why the real image is used [29].

## 3.2 Spatial Mapping

Spatial mapping and depth cameras are used in the newest AR systems to create a 3D representation of the surrounding environment, enabling the accurate placement of virtual items in the environment [35], [37]. A device's ability to map the environment around it is crucial to allowing various situations such as placement, occlusion, physics, and navigation. The spatial mapping capabilities of the HoloLens give sufficient precision to enable developers to build a lifelike MR experience [35], [38].

The first step in mapping the environment is depth sensing. For that, different methods can be used based on different hardware configurations. This may be accomplished in a variety of ways, depending on the hardware arrangement. Among these techniques are structured infrared light, stereovision, and ToF cameras. The structured light approach works by projecting a reference pattern onto an object, then the camera captures the distorted pattern where the depth is calculated based on the distortion level [39]. Stereo vision mimics how humans perceive depth using two different views where each point in the view has a different camera coordinate. Using the collinearity equation, the 3D position of each point is calculated.

ToF technology is of particular importance due to its widespread usage and the fact that it is being utilized to assess depth by devices such as the Microsoft Kinect and the HoloLens, among others. For the technology to operate, the items' reflective characteristics must be considered. It calculates the depth using the known speed of light by measuring the time it takes for a photon to reflect on the device's sensors [39].

Further mesh processing is typically required to identify typical forms of surface defects and to filter, delete, or amend spatial mapping data as necessary. Processing includes hole filling caused by dark material that fails to scan or smooth the noisy mesh surfaces [37].

On mobile platforms, building the map of the environment and localizing the device position is required when moving in an unknown environment, and it's typically performed simultaneously, which is known as the Simultaneous Localization And Mapping (SLAM) problem. To enable AR systems to function, the SLAM algorithm needs to run in real-time and minimize drift [40].

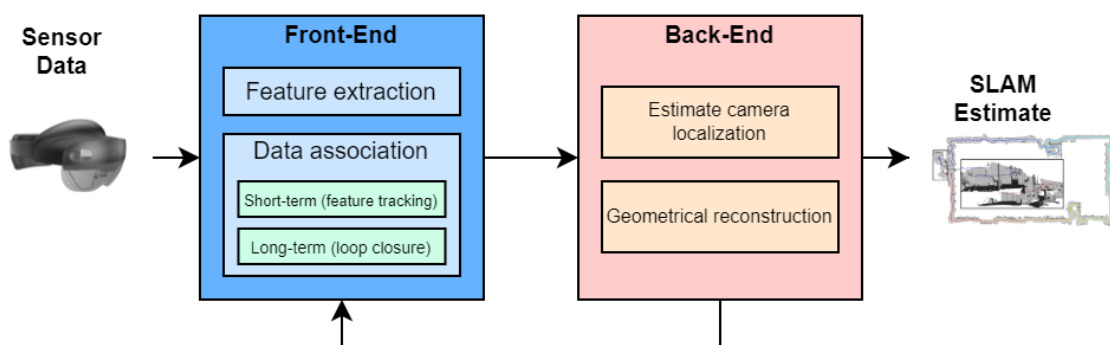


Figure 3-1: SLAM algorithm overview, based on [41].

As shown in Figure 3-1, the SLAM algorithm consists of four parts. The sensor data set comprises the camera frames themselves, as well as IMU and depth data. The front-end is the part that takes the raw sensor data and turns it into immediate representation, such as constraints in the optimization problem or a probability distribution about the location of keypoints derived from sensor data [41].

The back-end part takes the intermediate representation and solves the underlying state estimation or optimization problem, and the result will be an estimation of the location of the environment objects and platform position in that environment. The three categories of approaches for the back-end are: Extended Kalman filter, particle filter, or least square graph-based SLAM. Nowadays, graph-based SLAM is the most used algorithm [41], where a graph is used to represent the variables and the relations between them.

In HoloLens, visual-inertial SLAM (viSLAM) is utilized to combine visual and IMU data, which is processed separately to track the pose resulting in a loose coupling solution.

### 3.3 Coordinate Systems

Cartesian coordinate systems are used in 3D design software to define the position and rotation of 3D objects. Depending on how the X, Y, and Z axes are defined, several systems can be used. Two of the most common systems are left-handed or right-handed Cartesian coordinate systems [42]. In both coordinate systems, the positive X-axis points to the right, whereas the positive Y-axis points upward. The distinction is whether the positive Z-axis is facing the user or away from the user.

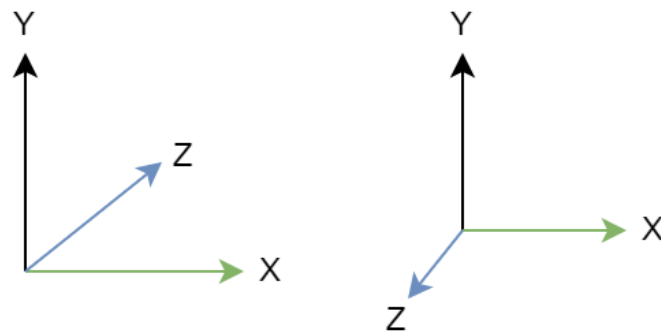


Figure 3-2: Cartesian coordinate systems. (*left*) Left-handed system. (*right*) Right-handed system.

The Unity game engine uses a left-handed coordinate system, where the coordinate values are expressed in meters. This means that objects separated by two units on any of the axes will appear to be separated by two meters when rendered. [42].

In AR HMDs, the digital content can be placed in a specific location in the real-world environment, which is known as world-locked content. On the other hand, head-locked allows the content to be rendered with the same view as the user. However, Microsoft doesn't recommend using head-locked content and suggests attaching the holograms to the user using custom solvers [42]. Solvers are modules that calculate the position and orientation of an item concerning the user's head location. As the result, the content can be placed with a fixed position of the user's head.



### 3.4 Spatial Anchors

Since the AR application and the real-world use different coordinate systems, to render AR contents in their true position relative to the real-world coordinate system, a reference point called a spatial anchor should be defined [43]. A Spatial Anchor is defined as a keypoint in the environment where the system tracks over time. Using the tracking methods mentioned in section 3.1, different keypoints with significant changes in one or more image properties can be detected.

As shown in Figure 3-3, the green points represent extracted keypoints and the white points represent the Spatial Anchor, where the spatial relationship between keypoints and the Spatial Anchor is known.

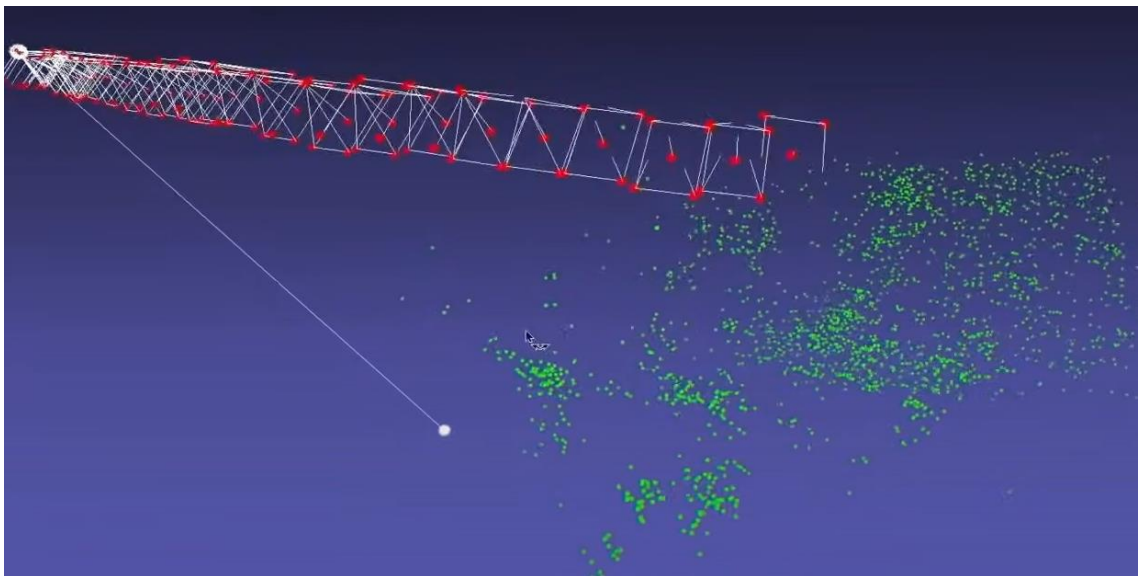


Figure 3-3: Relationship between Spatial Anchor and keypoints [44].

As described in 3.2, HoloLens runs the SLAM algorithm continuously and detects keypoints. To locate each spatial anchor, the detected keypoints will be compared to the reference Spatial Anchors information. Coarse relocalization can be done using Wi-Fi signals to filter spatial anchors based on their relevant location, which reduces the search-space radius to approximately 50-100m [45].

To be able to persist the AR experience during different application sessions, Spatial Anchors can be saved locally on the HoloLens.

The digital objects may be directly associated with the Spatial Anchors by specifying a local coordinate system for each anchor, with each anchor having an adjustable coordinate system

dependent on the coordinate systems of the other anchors [43]. The advantage is that the digital object will be rendered in the same position regardless of the user's location.

In general, if the digital content is placed more than three meters apart from the Spatial Anchor, lever-arm effects may lead to significant positional inaccuracies proportional to their distance from the origin [43]. Section 5.1 presents a possible approach that involves aligning the whole 3D scene with the real-world surroundings using Spatial Anchors that are placed automatically.

### 3.5 Shaders

Shaders are used in game engines to control how pixels are shown on the screen. Specifically, shaders, according to the official Unity documentation, are short scripts that include the mathematical computation and method for computing the color of each pixel produced, depending on the lighting input and material setup, among other things [46]. Such calculations are often performed on the GPU to reduce the amount of time necessary for rendering.

One of the three entities that play a part in the rendering process of the Unity game engine is represented in Figure 3-4. 3D models are a set of 3D coordinate vertices, which are used to construct the model. They are joined to form triangles when they are connected. Additionally, each vertex may have a few additional pieces of information, such as color, normal, and UV texturing map [47].

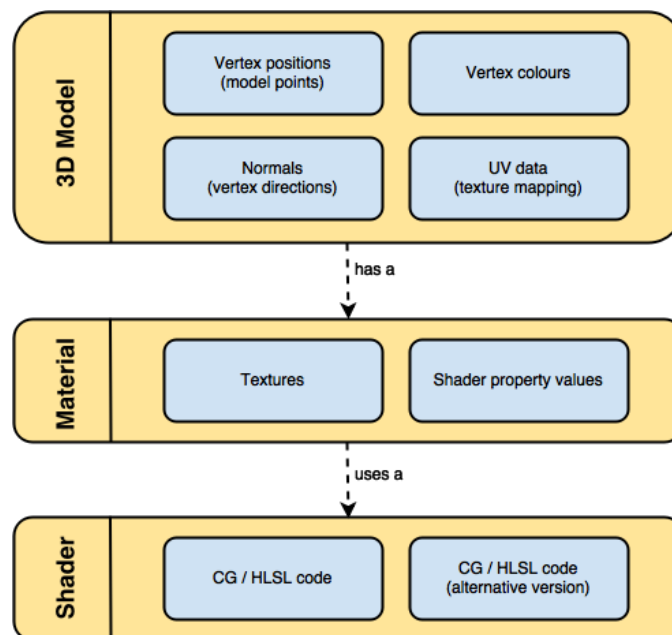


Figure 3-4: The relationship between 3D models, shader, and materials in Unity3D [47].

Models cannot be rendered until they have a material applied to them. Materials are containers that hold a shader as well as the values for the shader's attributes. Separate materials may thus share the same shader while supplying it with data from different sources.

Shaders in Unity are written in the High-Level Shading Language and are classified as surface shaders or fragment and vertex shaders [46]. Shader scripts often include a section called Properties that allows for the declaration and access of various variables from the inspector. The SubShader part includes the Shader code, which is run for each pixel in the image.

Vertex and fragment shaders operate like how the GPU draws triangles. The model's geometry is initially passed via a function that modifies its vertices. Following that, each triangle is passed through another algorithm that determines the final color value for each pixel [47]. The essential difference between fragment and vertex shaders and surface shaders is that the latter has no semantics for physical properties, which makes it more suitable for non-realistic materials.

## 4 DATA AND RESOURCES

This chapter begins with a description of the data source, followed by a detailed discussion of the data preparation process of this study. The second section provides an overview of the software that was used, including the AR SDKs and code compilation tools. The last section discusses the platform used and its key characteristics.

### 4.1 Data and Data Preparation

The available real-world representation consists of a 3D building model in the CityGML format. CityGML is an OGC open-source standard format for storing and exchanging 3D city models. In CityGML version 1.0 and version 2.0, the buildings are divided into multi-scale semantic representations based on Level of Detail (LOD), which refers to the model's resemblance to its real-world version and has consequences for the model's usability [48]. Figure 4-1 shows the five classifications of the building models based on LOD.



Figure 4-1: CityGML 2.0 building model classification based on LOD [48].

LOD1 represents the building's footprint solely, with no information regarding its height. By using primitive roof shapes, LOD1 approximates the height of the buildings. On the other hand, LOD2 provides more information on the roof shape. LOD3 and LOD4 are focused on providing information about the façade and interior, respectively [48].

The LOD1 model is provided for the study area and is sufficient to function as an occlusion mask since the observer is looking from the ground and no roof information is required [1], [15]. The model includes the whole city of Stuttgart and is provided in CityGML JSON format, thus requiring further data processing to use inside the game engine.

The FME workbench pipeline shown in Figure 4-2 was used to extract the buildings in the study area using an attribute filter based on the block number, and then an OBJ writer was used to generate a mesh output.

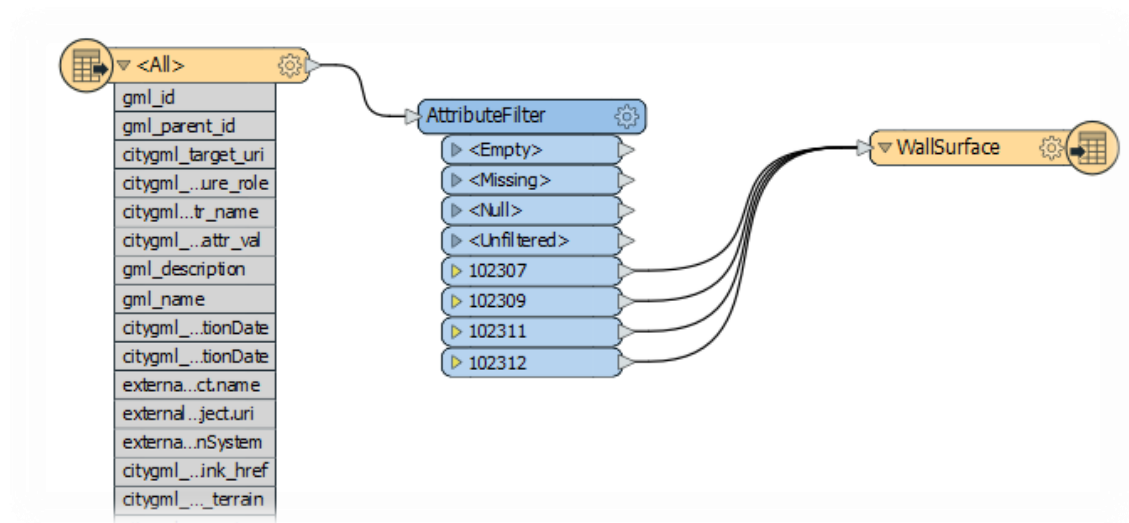


Figure 4-2: FME pipeline to convert CityGML to OBJ mesh.

The building model is constructed using UTM projected coordinates, with the Z value representing the height of the building. As previously established in section 3.3, game engines define height by using the Y value in their representations. Intending to use the building model in the game engine, the OBJ model was rotated and moved to a local coordinate system with an origin at 0,0,0 as seen in Figure 4-3.

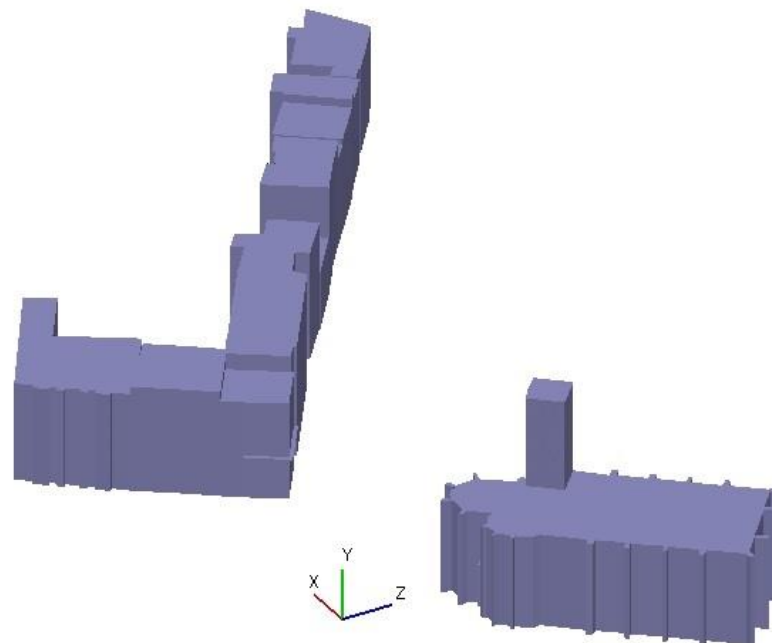


Figure 4-3: 3D mesh building model of the study area.

## 4.2 Software

The AR application was developed using Unity, which is a cross-platform game engine capable of developing 3D and 2D games, as well as interactive simulations and other experiences [49]. Unity is freeware and can be used free of charge for non-commercial uses, which makes the methodology applicable in different study areas. The application was developed using the Unity 2020.3 long-term support version.

MRTK provided a cross-platform input system, foundational components, and common building blocks for spatial interactions, for which it was used as an AR SDK. MRTK also leverages access to different platform capabilities, including hand tracking and gaze. MRTK provided direct access to Spatial Anchor and World Looking Tool, both of which were used to place and maintain the location of the augmented content. MRTK version 2.7.2 was used to develop the application.

Vuforia is an AR SDK that specializes in the real-time detection and tracking of 2D and 3D targets and can support IMU data integration for extended tracking. Therefore, the digital content can be shown even if the target is lost. Vuforia version 10.3 was utilized, and Area Target tracking was tested. However, it was not used in the final development of the AR application because of technical limitations.

Visual Studio 2019 was used as a script editor and for building and deploying the application to the platform. Thus, all components were running on Windows 10.

## 4.3 Hardware

The Microsoft HoloLens is a self-contained, wearable AR device that allows the user to display interactive 3D holograms in the immediate vicinity with the support of a natural user interface. Therefore, HoloLens works without a smartphone or an additional computer. HoloLens 2.0, which was released in November 2019, was used as the main AR platform in this research.

The HoloLens has two processors. The first one is the Holographic Processing Unit 2.0 which contains a Deep Neural Network core, and the processor is dedicated to real-time computer vision tasks, including head, hand, and eye-tracking as well as the SLSM algorithm for spatial mapping. The second is the Qualcomm Snapdragon 850, which integrates an Adreno 630 GPU for 3D rendering and application processing [38], [50]. HoloLens has an IMU that contains an accelerometer, gyroscope, and magnetometer to determine linear acceleration and rotation along the three axes.

The HoloLens is equipped with a total of six cameras at the front shown in Figure 4-4: four grayscale cameras in two stereo pair configurations. The ToF camera has two separate modes: a long-throw for mapping the environment with a range of five meters and a short-throw with a range of one meter for hand tracking. The sixth camera is an RGB to capture images and videos. Table 4-1 outlines the characteristics of the HoloLens cameras.



Figure 4-4: HoloLens 2.0 camera configuration. (yellow) Grayscale tracking cameras. (Red) ToF camera. (Blue) RGB camera.

Table 4-1: HoloLens camera characteristics [50], [51]

Camera	Image Size [Pixels]	Frame Rate [FPS]	Format
Grayscale	$640 \times 480$	30	8-bit
Long Throw	$320 \times 288$	5	16-bit
Short Throw	$512 \times 512$	45	16-bit
RGB	8-MP stills, 1080p video	30	8-bit

Also, the HoloLens does not have a GPS or a GNSS sensor, so it depends entirely on computer vision to understand the surrounding environment and determine its location. This cannot be considered negative aspect, as previous research has shown that the accuracy of GPS is fundamentally insufficient to create an AR [1], [5].

The HoloLens lenses are transparent, allowing the user to see straight through them. They do, however, have a series of very fine, invisible grooves that guide the virtual images created by the application into the user's eyes. The display has a brightness of 500 nits and can show a resolution of up to 2K per eye [51]. Rendering different perspectives for each eye gives the impression that virtual content is in various places and distances around the environment. The iris sensor helps track the location of the user's eye and adjusts the screen according to eye measurements for a 3D experience.

The HoloLens design includes front and back cushioning as well as an overhead strap to make wearing it more comfortable, which is important considering how heavy it is at 535 grams [51], [52].

For the Vuforia SDK tracking to work properly, it was essential to scan the study area using the iPhone 12 Pro's LiDAR capabilities to create an Area Target that could be used for testing. iPhone LiDAR has a range of five meters that is comparable to that of the HoloLens.



## 5 METHODOLOGY

The first section of this chapter describes the proposed method for this thesis work and overall workflow. The second section discusses the evaluation criteria used to evaluate the proposed method in terms of performance and users' feedback.

### 5.1 Proposed Methodology

Based on a review of the literature on existing techniques and the state-of-the-art performances demonstrated by various tracking methods and occlusion handling techniques, this method proposes the use of Spatial Anchors to align the 3D scene with the real-world environment.

Figure 5-1 gives an overview of the workflow of the proposed method, where the scanned data are used as a base layer to place different design objects and the building model according to the Unity local coordinate system. Spatial mapping is done in real-time on the HoloLens alongside Spatial Anchor detection and transformation optimization.

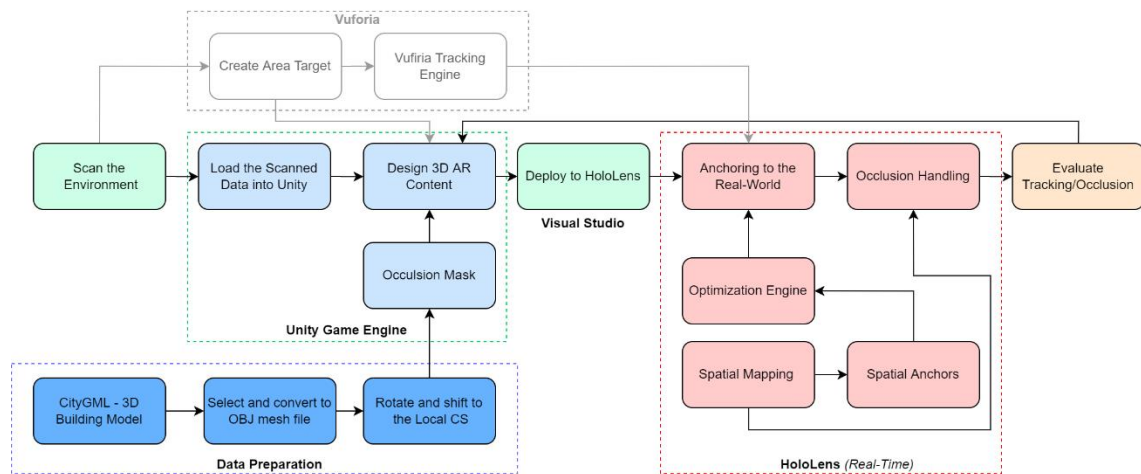


Figure 5-1: An overview of the workflow of the proposed method.

The issue of aligning digital content with the physical environment is split into two parts in this thesis. To begin with, a test environment is provided to assess two different tracking methods using data from various sensors. The second part is to find the best method and implement it to run on the HoloLens headset. This is intended to run in real-time to align the 3D content with its desired location in the real world. Thus, accuracy, consistency, and computation time are considered when evaluating the methods on the test platform.

The Spatial Anchor concept described in Section 3.4, is used to define a reference point and its relationship to other keypoints detected. Utilizing a single anchor for each distinct 3D object allows for positional stability, but at the expense of self-consistency within the scene. Instead, a set of anchors that were generated and distributed automatically was used. Based on those anchors, the entire 3D scene was aligned with the actual world, after which a transformation adjustment was applied to the camera component.

The Spatial Anchor and current head tracking data acted as a dynamic input for the World Looking Tool and was referred to as the “Spongy State” at the start of the application, where the origin point of the local coordinate system is based on the head pose. Following, an optimization engine aligned the scene with the real world by applying rotation and translation since the scale is consistent. To achieve persistence across different application sessions, the calculated transformation was saved in a “Frozen State”.

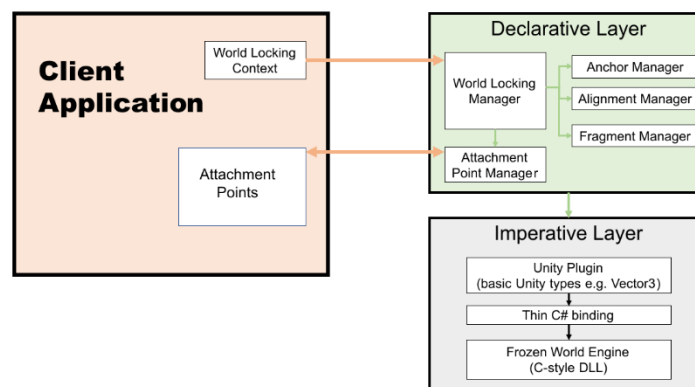


Figure 5-2: World Locking Tools architecture [53].

Figure 5-2 summarizes the World Locking Tools for Unity architecture, where the client application initiates the World Locking manager, which in its turn starts building the Spatial Anchors through the Anchor manager. The Alignment manager calculates the transformation needed to generate a frozen state.

Since the World Locking Tools are a component of the MRTK SDK, they make use of keypoints that are already in use in the SLAM algorithm, allowing the tools to operate in real-time and without the need for a pre-mapping of the environment. On the other hand, the Vuforia Area Target requires mapping the environment first. A scan of the real world was made using the iPhone 12 Pro to generate a spatial map including a point cloud with X, Y, and Z coordinates as well as an RGB value for each point. Also, three normal directions are calculated.

Furthermore, this thesis puts forward the idea of using the 3D building model as a custom shader to enrich the spatial map and create the occlusion effect. Section 4.1 describes how the data was converted into 3D mesh where it was used in the Unity game engine. The occlusion shader was used to determine how to render each pixel based on the line-of-sight between the observer and the augmented content.

Due to the evaluation findings, it is important to note that the Vuforia Area Target technique was not utilized in the final implementation, but it is shown in the workflow to demonstrate how it can be used.

## 5.2 Evaluation Criteria

The visual effect of the occlusion screening was shown via screenshots, and the occlusion effect's impact was evaluated through user feedback. Furthermore, the suggested methodology was assessed using a set of performance indicators. Since pixel-perfect occlusion is not necessary to acquire the intended occlusion effect [1], no statistical analysis was performed.

### 5.2.1 Performance

The performance of the proposed method was evaluated, taking into consideration the number of Frame Per Second (FPS) that the application rendered the augmented content. The number of FPS is a crucial element for user comfort [52], as the higher the FPS can cause less blinking frequency, which in turn causes less stress for the user [54]. A judder problem can occur as well in cases of low FPS rendering [55], which can cause screen tearing where multiple frames are rendered in the same view.

It's critical to evaluate the speed and robustness of the tracking system and the position alignment of the augmented content. Since the augmented content that is fixed in the environment should maintain its position regarding the user's movement. However, a drift of augmented content location might happen if it's placed outside of the mapped environment or far away from the spatial anchor. While the whole scene is anchored and not an individual object, this problem is not expected to happen. Jitter can be seen by users as a high-frequency movement of augmented content, which may occur as environment tracking fails.

The HoloLens has a built-in hologram stability pipeline [55] that gives good results in maintaining the stability of the augmented content [56]. An eye calibration is necessary to avoid the swim effect where the whole augmented content appears floating while the user moves his head.

### 5.2.2 Users' Study

A user study was conducted to gather feedback regarding the augmented content. A demo session for users begins with a brief introduction to the research and its overall goal so that the users are aware of what they are looking for in terms of evaluation and may prepare accordingly. Following that, the participants were free to move around the study area, where they can enable and disable various occlusion options as they pleased. The users were given a questionnaire that consisted of five sections and a total of twelve questions. Likert [57] options were used to answer all questions, on a scale from 1 to 5, with 1 indicating "Strongly Disagree" and 5 representing "Strongly Agree".

In the first section, the participants were asked about their prior knowledge of AR technology and its application. Then they were asked whether they were acquainted with HMDs. The second section is primarily concerned with the overall impression and experience immersiveness of the HoloLens, as well as the users' ability to move freely while wearing the device. The third section was concerned with determining the position stability of the augmented content. The fourth section focuses on the occlusion effect based on the in-device spatial map or based on the 3D city model as an occlusion mask. The fifth and last element is concerned with the safety of the participants and if they experienced any side effects, such as eye strain or any uncomfortable feelings during or after the experience.

## 6 IMPLEMENTATION

This chapter is concerned with the implementation aspects of the methodology discussed in Chapter 5, including a description of the project setup and an explanation of the custom shader that was used, as well as the design of the 3D content itself

### 6.1 Project Setup

Setting up a Unity project begins with selecting the target platform, which may be either Android, iOS, Universal Windows Platform (UWP), or another platform. It is also critical to choose the suitable Architecture as well as the correct Target Device for your application. The HoloLens and the ARM64 processor were chosen for this purpose. After that, the MRTK AR SDK was loaded into the game engine as a Unity package, which contains all the components and scripts needed. MRTK was able to access the HoloLens sensors since a variety of capabilities were enabled, including access to the camera, microphone, internet client, and spatial perception.

A Unity project is divided into scenes, which include all the application's components, including all Game Objects, which serve as a container for other components. While using MRTK, the scene must be configured in such a manner that there is a single main camera that serves as a user viewport. Additionally, a global light source called Directional Light was added to illuminate the whole scene. The main camera needs to clear the screen to black color before rendering the scene, where black represents no color values. In other words, black pixels indicate that the user is viewing the real-world surroundings and that no content is being shown on the screen. To make the application suitable for outdoor usage and beyond the HoloLens' five-meter scanning limit, the experience option in MRTK was set to World experience.

### 6.2 Spatial Observer

As discussed in section 3.2, HoloLens can scan and reconstruct the surrounding environment, and it's possible to use the spatial map as a mesh. For that, Spatial Awareness was activated, and a Spatial Observer was added to the project setup to pass the scanned mesh from HoloLens to the application where it can be used. Figure 6-1 shows an interior test scene comprised of two rooms and a small corridor, while Figure 6-2 illustrates the same spatial map mesh overlaid over the real environment.

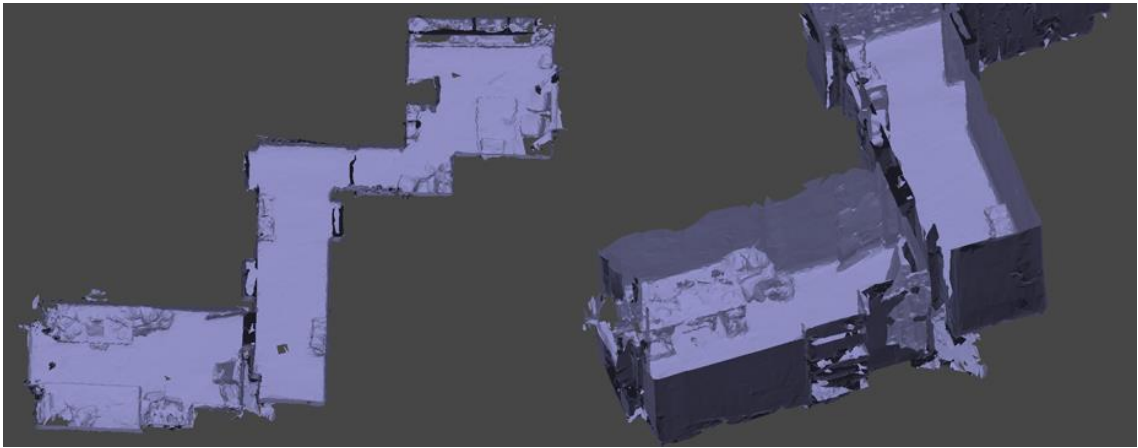


Figure 6-1: Spatial map of an indoor environment captured by the HoloLens. The ceiling has been sliced to improve visibility.

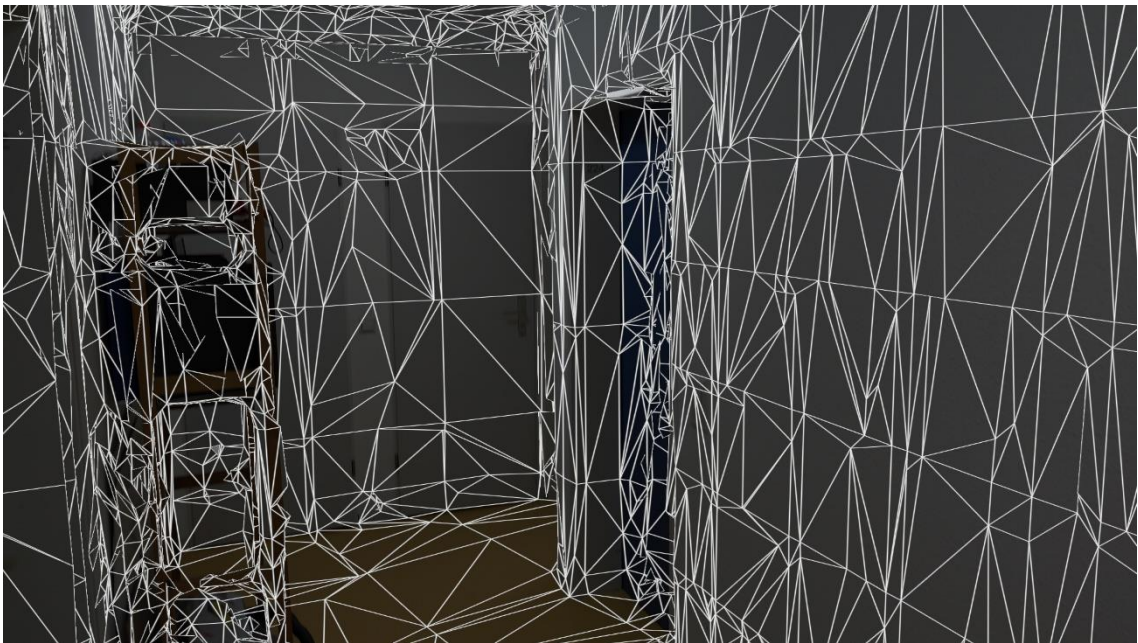


Figure 6-2: Spatial map mesh superimposed over the real environment scene.

The scanning update interval, which refers to the amount of time the HoloLens takes between two scanning pulses, is controlled by the Spatial Observer and was set to two seconds. The LOD of the spatial mesh is controlled using the Spatial Observer. Depending on the intended usage, different LODs can be requested during the mesh processing. The LOD indicates how many polygons per cubic meter are generated, and for occlusion effects, it's important to get a fine mesh representation of the environment.

The different display options determine how the mesh is rendered on the screen, with different materials and shaders assigned to each option. The Visible option renders a transparent polygon



with white borders. On the other hand, using the Occlusion option with an occlusion shader can be used to achieve the occlusion effect. Both behaviors were controlled using a script and the dashboard mentioned in section 6.4.

## 6.3 Anchoring

To render the 3D content to its desired location, Spatial Anchors were used and automatically distributed in real-time as described in section 3.4. For that, World Locking Tools were added to the Unity scene and the main camera game object was stored in it. As shown in Figure 5-2, the World Locking Tools script initiates the World Locking Manager, which in turn takes control of placing the spatial anchors and performing the camera transformation needed.

Scale inaccuracy is another critical element to consider. This issue causes users to see their model as being larger than it was intended to be. This misalignment occurs because the distance traveled in head-tracked space tends to vary from the distance traveled in a real-world environment by the inaccuracy of  $\pm 10\%$  [53].

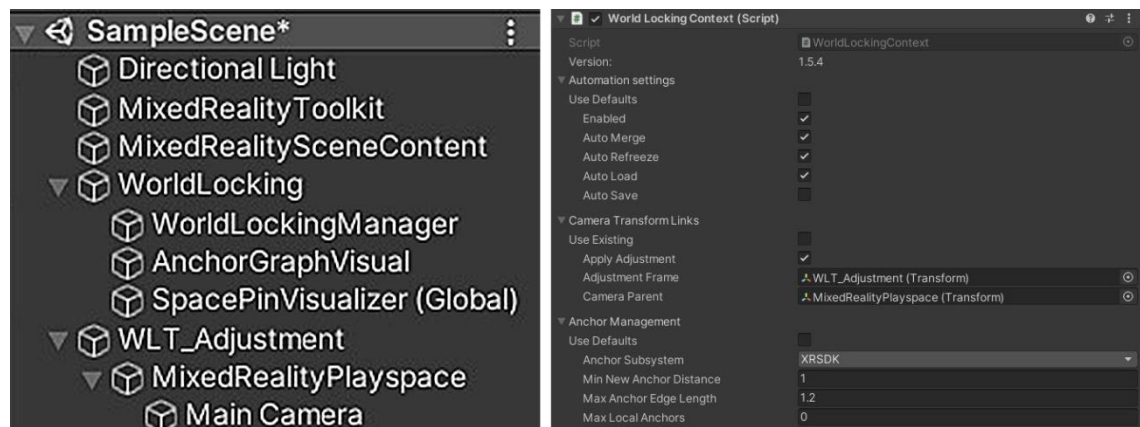


Figure 6-3: World Locking Tools. (left) Game objects hierarchy. (right) World Locking Tools configurations.

Figure 6-3 illustrates how the main camera object is incorporated inside a World Locking Tools game object and used as input by the Optimization Engine through the World Locking Manager script. Additionally, there is no restriction on the number of Spatial Anchors, and the minimum distance between anchors has been set at 1.2 meters to allow for the addition of extra anchors as the user moves around the environment. The save function allows the application to save the current Frozen State at application run-time, where it can be loaded at the startup of different sessions using the load function. The process can be done automatically without user interaction or manually using the in-application dashboard shown in Figure 6-4.



Figure 6-4: World Locking Tools in-application dashboard.

Other information related to the current number of Spatial Anchors, linear and angular deviation, and visualization of the position of the Spatial Anchors and their relationship to one another can be displayed using the above-mentioned dashboard. Additionally, the Spongy position represents the user's initial location when the application is launched.

The additional in-application dashboard shown in Figure 6-5 was used to control a script described in Annex I. to fine-tune the alignment, including shifts along the 3-axes and rotation around the Y-axis.

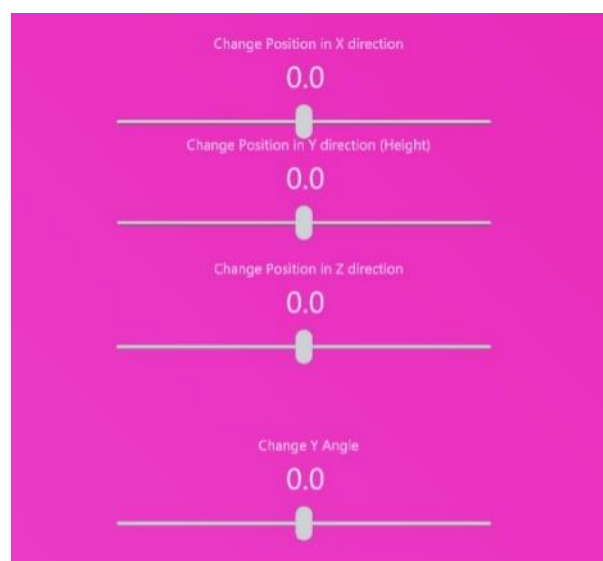


Figure 6-5: Alignment fine-tuning dashboard.



## 6.4 Occlusion Shader

As discussed in section 3.5, the occlusion shader calculates and determines how each pixel is rendered on the screen. The shader is wrapped in a material that can be assigned to the 3D building model to act as an occlusion mask. The full shader code is shown in Annex II.

As shown in lines 7-11, to render the occlusion mask before all other meshes in the scene, a Queue value is set to one unit lower than the geometry value where the geometry value is always 2000 and represents the priority to render solid objects. ZWrite is enabled to write the pixels from the occlusion mask to the depth buffer. The ZTest is set to render the geometry at the front and not draw any pixels behind it from other objects. In other words, all objects behind the occlusion mask will not be rendered. Moreover, the ColorMask is set to zero, enabling the shader to modify all three RGB values and the alpha channel as well, which represents transparency.

Lines 15-19 specify the vertex and fragment functions used. While lines 21-31 define the needed structure to hold the input and output position information. The POSITION parameter is used to pass the 3D coordinates of each pixel, which are stored in 32bit, whereas SV\_POSITIO stores the object screen coordinates in addition to the Z value returned by the ZTest.

Lines 33-41 represent the main function used to convert vertex object-space 3D coordinates to screen 2D coordinates, then the function fixed4 in lines 43-46 returns a black color value for each vertex when it's rendered on screen.

The HoloLens uses additive color mixing. As a result, the color black appears transparent. However, the geometry of the building model which acts as an occlusion mask is drawn in the scene, any object hidden behind it will be occluded.

To control the occlusion effect a user menu was added, where the user can enable and disable the occlusion screening as well as visualize the occlusion mask in visible colors. Additionally, occlusion based on the in-device spatial map can be controlled. Annex III shows the script related to the menu. The solver concept was implemented in the user menu as well, enabling the menu remains at a fixed position of the observer location.



Figure 6-6: User menu to control occlusion screening behavior.

## 6.5 Designing the 3D Content

In order to fully represent a proposed draft given by the city of Stuttgart to redesign the study area, 3D content was built. The original idea was to demolish the current Züblin parking garage and replace it with a new structure that could be used for the following purposes 60 percent multi-story parking garage, 20 percent office building, and 20 percent residential space. As a bonus, green space around the structure has been established.

The suggested concept draft for the redesign of the Züblin parking garage is seen in Figure 6-7.



Figure 6-7: Proposed concept draft to redesign Züblin parking garage area.

In response to that, the 3D model of a proposed building shown in Figure 6-8 was designed and integrated into the application.



Figure 6-8: 3D model of a proposed building to replace the Züblin parking garage.

The surrounding green area was produced using a plane object with a Texture shader of a grass texture repeated over the whole scene. The texture shader gives the advantage of adding realistic details to objects by overlaying 2D images and at the same time reducing the computation time needed [46]. The Terrain Editor from Unity was tested as well since it's capable of creating landscape scenes with realistic grass movement and lighting. Other bushes were designed based on 3D models with a leaf texture. The Unity Tree Editor provided more control of different tree parts e.g., tree stem, leaf movements.

Additionally, a grass hedge inspired by the hedge present in the surroundings of the study area was added to the 3D design to establish the impression of a car-free environment. This, in turn, contributes to the overall immersive experience given by the design. The 3D-created grass hedge and bushes are illustrated in Figure 6-9.

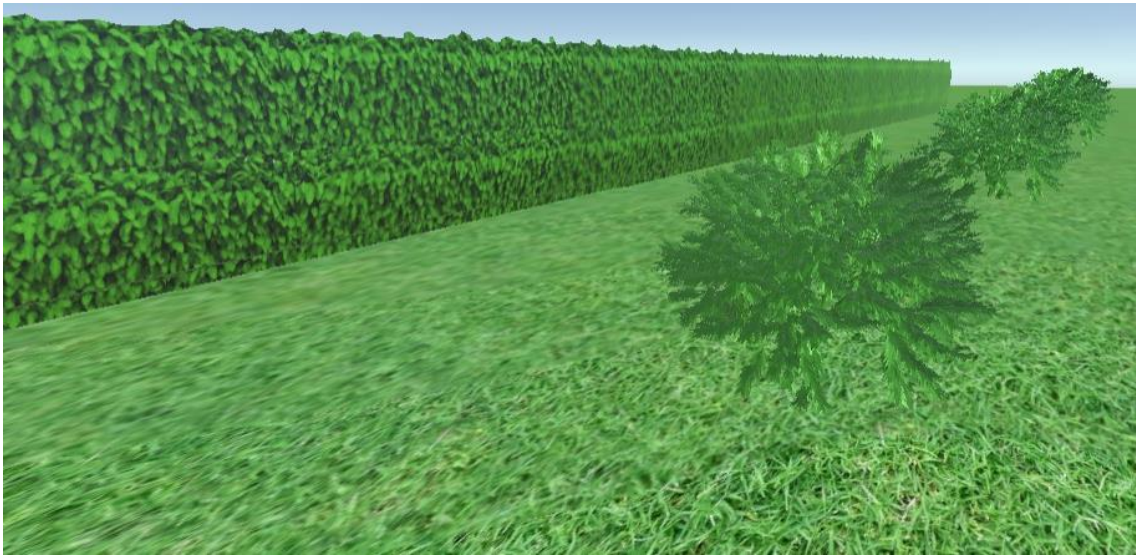


Figure 6-9: The 3D-designed grass hedge and bushes.

## 6.6 Compiling and Deploying the Application

The initial stage in developing the application is to use the unity build function, which is where it is critical to include the scene that contains the design. During the project setup process, the configuration options for the build were also defined. Following that, all the packages and assets will be bundled together with a Visual Studio solution file for easy distribution. In order to compile the solution and build an executable file that can be deployed to the HoloLens, Visual Studio must be used in conjunction with the other required components. For the solution to be compliant, the architecture used while designing the application must match that used when building the solution. The image below depicts the settings and procedures that must be followed to compile and deploy the application. It is also recommended that the developers' options in the HoloLens be activated to transfer the new application to the device.



## 7 RESULTS AND EVALUATION

This chapter presents the results of the proposed methodology and design and implementation discussed in Chapter 5 and Chapter 6. Its first section consists of the output of the tracking system and anchoring methods. The second section illustrates the occlusion effect based on the in-device spatial map and the 3D building model and provides a visual comparison of both. The third section evaluates the performance of the developed application. The fourth and final section reports the feedback gathered from the users' study.

### 7.1 Tracking System and Anchoring

Anchoring the augmented content based on the Vuforia Area Target concept and Spatial Anchors with World Locking Tools was tested. Three Area Targets totaling 832 square meters were scanned, with the scan duration limited to five minutes by the Area Target generation app to avoid overheating the iPhone. The Vuforia engine's behavior was unstable, and the localization time varied between two seconds and two minutes. In some tests, the engine was unable to localize the device position at all. Figure 7-1 shows a successful experiment and a comparison that shows the location of the test target in the game engine and the real environment tracked and localized using Vuforia Area Target.

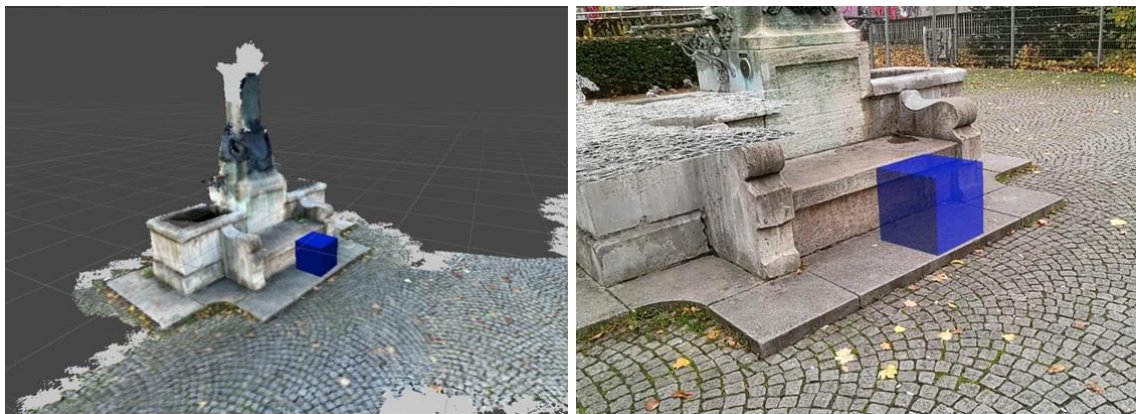


Figure 7-1: Localizing a test object using Vuforia Area Target. *(left)* Game engine view.  
*(right)* Real-environment implementation.

On the other hand, using World Locking Tools does not require a pre-scan as it simultaneously maps the environment and localizes the device using the in-device spatial map. Additionally, the area was not bounded, as new Spatial Anchors were added as the user moved. The localization of the device position and the 3D scene coordinate transformation was done in real-time.

Figure 7-2 shows a visualization of the Spatial Anchors distributed in the study area and how they are connected in a form of a network.

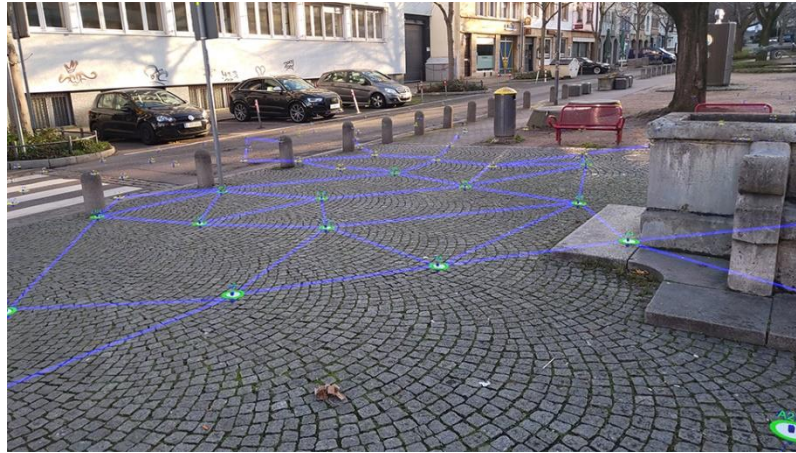


Figure 7-2: Distributed Spatial Anchors in the study area.

The number of Spatial Anchors utilized during different test sessions are shown in Table 7-1, as well as the maximum linear and angular deviations, which are calculated internally based on the location of all Spatial Anchors.

Table 7-1: Number of Spatial Anchors, linear and angular deviation during test sessions.

Test Session	Session 1	Session 2	Session 3
Number of Spatial Anchors	63	127	271
Maximum Linear Deviation	34 mm	33 mm	22 mm
Maximum Angular Deviation	0.6830 °	0.3323 °	0.2807 °

The linear deviation represents the distance between the currently detected Spatial Anchors in the “Spongy State” and the reference points corresponding to them in the “Frozen State”. The lower the value indicates finer alignment. The angular deviation indicates the angle difference between the user location in comparison to the Spatial Anchors in both “Spongy State” and “Frozen State”.

Based on the above-mentioned Spatial Anchors, the application was able to render the augmented contents according to their desired location. Figure 7-3 shows the study area before and after overlaying the augmented content and how it was possible to create a visualization that represented a car-free environment.



Figure 7-3: Study area before (*left*) and after (*right*) superimposing the augmented content.

## 7.2 Occlusion Screening

As discussed in section 6.4 the geometry of the scanned spatial map and the 3D building model was used as an occlusion mask were given a custom material based on developed the occlusion shader. Figure 7-4 illustrates the used occlusion mask inside the Unity game engine.

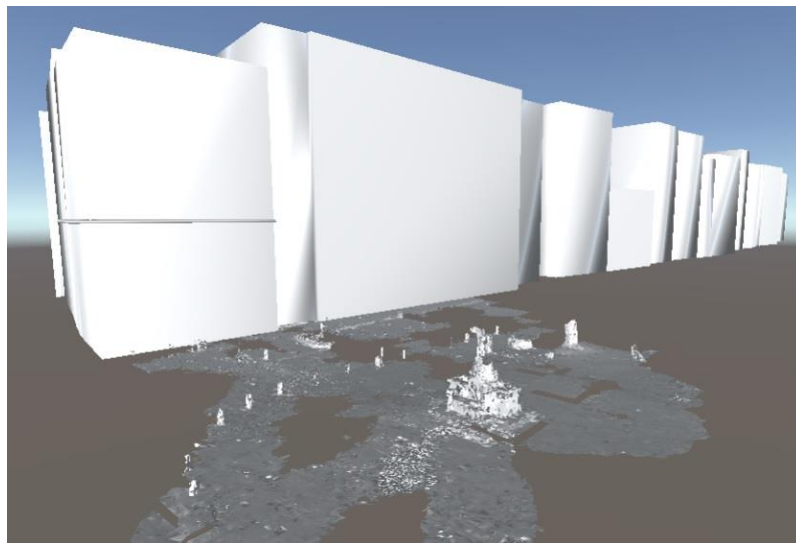


Figure 7-4: Occlusion mask based on scanned spatial map and 3D building model.

The developed occlusion screening's visual effect based on the in-device scanned spatial map is shown in Figure 7-5, wherein the left side shows how the 3D design appears when the occlusion effect was disabled. The grass and the grass hedge appear in front of the statue and the tree even though they are supposed to be in the line-of-sight between the observer and the augmented content. In comparison to the right side where the occlusion effect was enabled, the 3D design was occluded based on the shape of the scanned real-environment objects.





Figure 7-5: Occlusion screening based on the spatial map. (*left*) Disabled. (*right*) Enabled.

Nevertheless, the occlusion mask based on the spatial map had flaws where it didn't align perfectly around the edges. Also, some parts were missing or were not scanned, which led to an incomplete occlusion effect. Some noises resulting from the incomplete scanning of the environment have been observed, particularly while pedestrians pass in front of the observer. Even after the pedestrians were no longer visible in the scene, it took time for the spatial map to update and exclude their masks. Also, the same noise behavior caused by the passing cars was noticed. Figure 7-6 shows the above-mentioned noises.



Figure 7-6: Noises during spatial map scanning caused by (*left*) moving pedestrians and (*right*) passing cars.

Figure 7-7 illustrates the 3D building model that was used as an occlusion mask. The left side shows how the 3D design appeared floating over the real building when the occlusion is disabled. While the right side represents how the scene looked when the occlusion screening was enabled. The occlusion mask aligns with the real-life equivalent as shown in Figure 7-8 and the augmented content occluded along the edge of the building. The occlusion mask appeared sharp around the edges since the geometry of the buildings was represented with straight meshes.

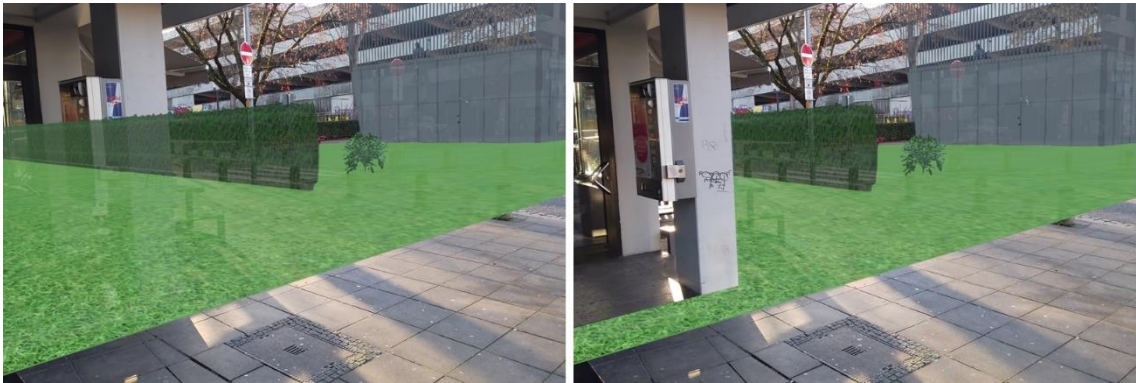


Figure 7-7: Occlusion screening based on the 3D building model. (*left*) Disabled.  
(*right*) Enabled.

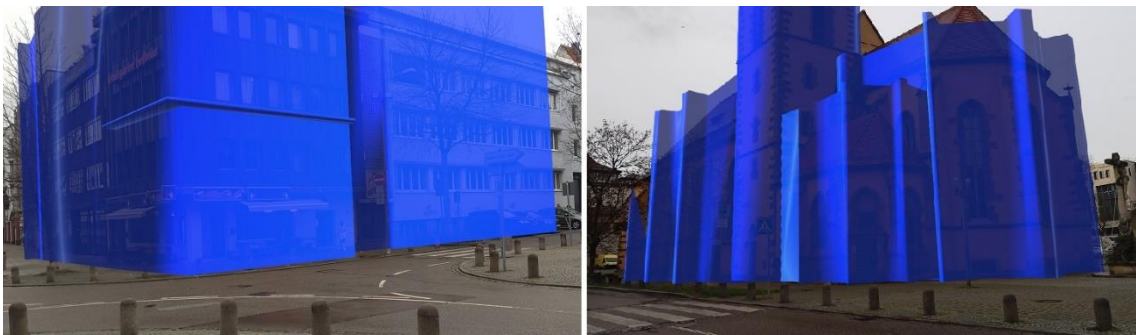


Figure 7-8: Alignment of the 3D building model occlusion mask with real environment.  
Rendered in blue color for better visualization.

Figure 7-9 provides a comparison of the occlusion screening effect based on the spatial map scanned by the device on the left and by using the 3D building model as an occlusion mask on the right side. It's noteworthy that HoloLens couldn't scan the glass facade of the building, resulting in an incomplete occlusion effect. Furthermore, if the observer stands outside of the scanning range, which is five meters, the occlusion based on the spatial map will not be effective. In opposition, using the spatial map as an occlusion mask, the augmented content was occluded on top of the tables beside the building.



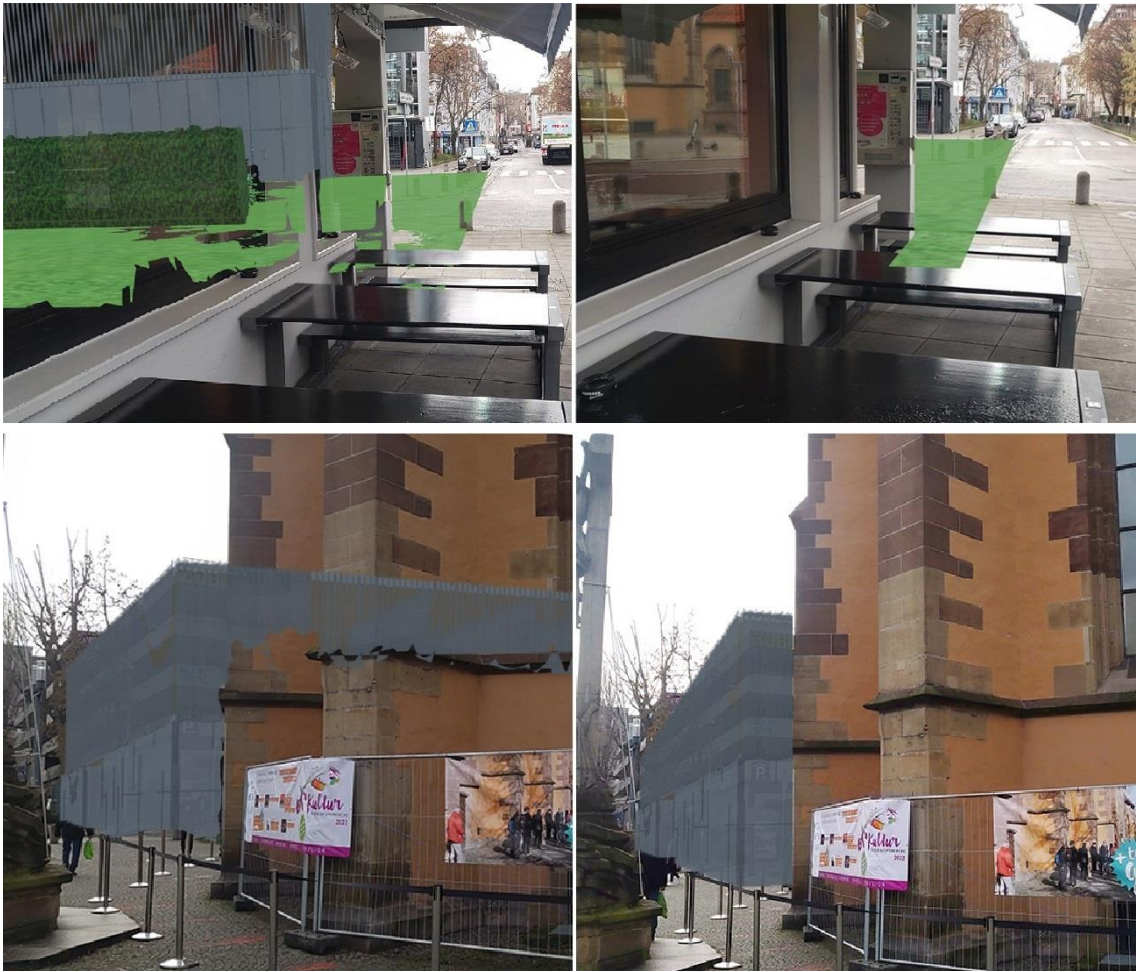


Figure 7-9: Comparison between occlusion screening based on (*left*) the spatial map and (*right*) the 3D building model.

### 7.3 Performance Evaluation

Two distinct 3D designs were used in the performance test, which focused on determining the FPS throughout five minutes of continuous application use. The first design represents the final implementation and consists of around 1.84 million triangles, most of which come from the detailed design of the parking garage building and trees. The second design replaces the grass texture shader with a 3D detailed grass model, which adds another 3.8 million triangles, bringing the total to 5.24 million triangles in total. Figure 7-10 shows the number of rendered frames over the test duration. Using the first design, the application was able to render at 60 FPS on average with 1-2 drop frames. On the other hand, using the second design, the application rendered at 34.2 FPS with a maximum of 45 FPS and a minimum of 20 FPS. As the dropping of the frames occurred in conjunction with the movement of the user or the rotation of his head and the change of the rendered scene.

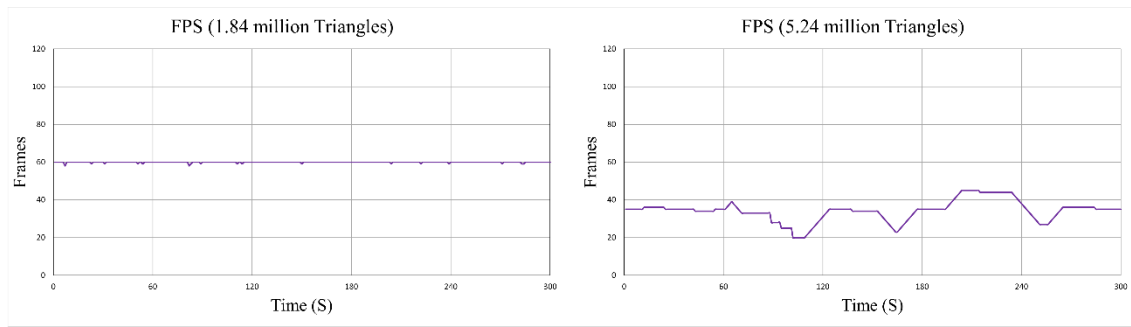


Figure 7-10: Relation between the number of triangles and rendered FPS.

Using the Unity Tree Editor and Terrain Editor, a similar trend of high-poly design occurred, with an average, maximum, and minimum FPS of 25.3, 29, and 10, respectively. It was also noticed that when using trees from Tree Editor, some areas, especially the stem, are rendered black, which turns transparent when using the application on the HoloLens, and therefore this part of the trees disappears.

## 7.4 Users' Study Feedback

As discussed in section 5.2.2, users' opinions were gathered through a demo session followed by a questionnaire, which is summarized in Figure 7-12. The experiment enrolled 14 participants from a variety of backgrounds and age groups. The first section elicited responses regarding participants' prior knowledge, with three responding with “excellent” knowledge of AR and two responding with “very good” knowledge. While most participants responded with “sufficient” prior knowledge, the remainder of the responses fell into the “fair” and “poor” categories. The average response for this question was 3.14 seconds, and the median response was 3. When asked how familiar they were with see-through HMDs, only two participants responded “excellent”, while the remaining responses were evenly distributed among the other options, resulting in an average of 2.86 and a median of 3. The participants' prior knowledge is summarized in Figure 7-11.

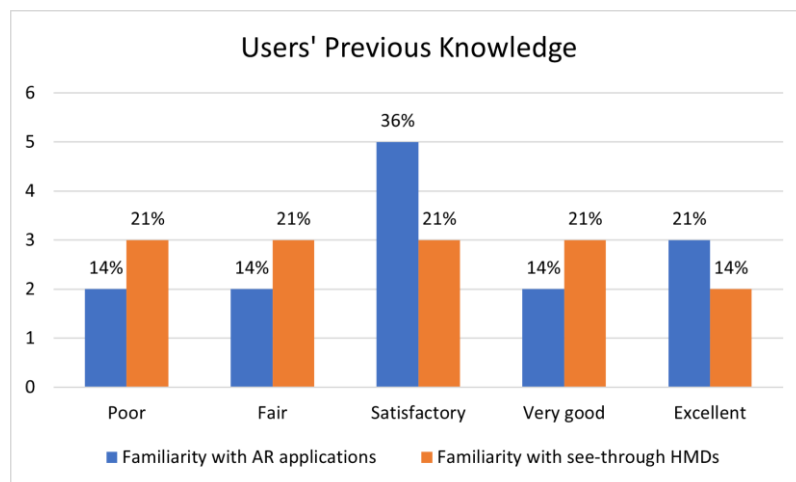


Figure 7-11: Participants' answers to the question about their prior knowledge.

Following that, participants were given a general statement regarding their overall impression of the developed application. Four participants “strongly agreed” that it was a positive impression, four participants “agreed,” and only two participants were “neutral” or “disagree”. The average and median answers to this statement were respectively 4.07 and 4. When participants were asked if the augmented reality experience was immersive, four responded “strongly agree” and eight responded “agree”. Only two respondents indicated that they were “neutral” or “disagree”. Both the average and median answers to this statement are 4. The next statement inquired whether the participants were free to move during the experience; three responded “strongly agree” and eight responded “agree”. Only two of the participants were “neutral”, and one was “disagree”. The average answer for this statement was 3.93 and the median was 4.

The following statement was about the tracking system, in which participants were asked if the augmented content remained stable in its position throughout the experience. Eight participants “strongly agree” and five participants “agree”, but only one participant was “neutral”. The average and median responses to this statement were respectively 4.5 and 5.

The occlusion effect was the main topic of the fourth section of the questionnaire. To begin, participants were required to state that the occlusion effect was sufficient for this application. The response to this statement was remarkable, with ten “strongly agree” and the remaining “agree”, resulting in an average of 4.71 and a median of 5. Then they asked if the occlusion effect enabled them to have a better depth perception were nine answered “strongly agree” and five answered “agree”. The average and median answers were 4.64 and 5 respectively.

The final section was related to the participants’ safety, with the first statement inquiring whether they were aware of the real environment during the experience. Five participants responded

“strongly agree”, five participants responded “agree”, three participants responded “neutral,” and only one participant responded “strongly disagree”. Both the average and median answers for this statement were 4. When participants were asked if they experienced any eye strain during or after the experience, ten responded “strongly disagree”, two responded “disagree”, and only two responded “neutral” or “strongly agree”. The average response was 1.57, while the median response was 1. The last statement was concerned if the participants had any uncomfortable feelings, ten responded “strongly disagree”, two responded “disagree”, and only two responded “neutral” or “agree”. The average response was 1.50, while the median response was 1.

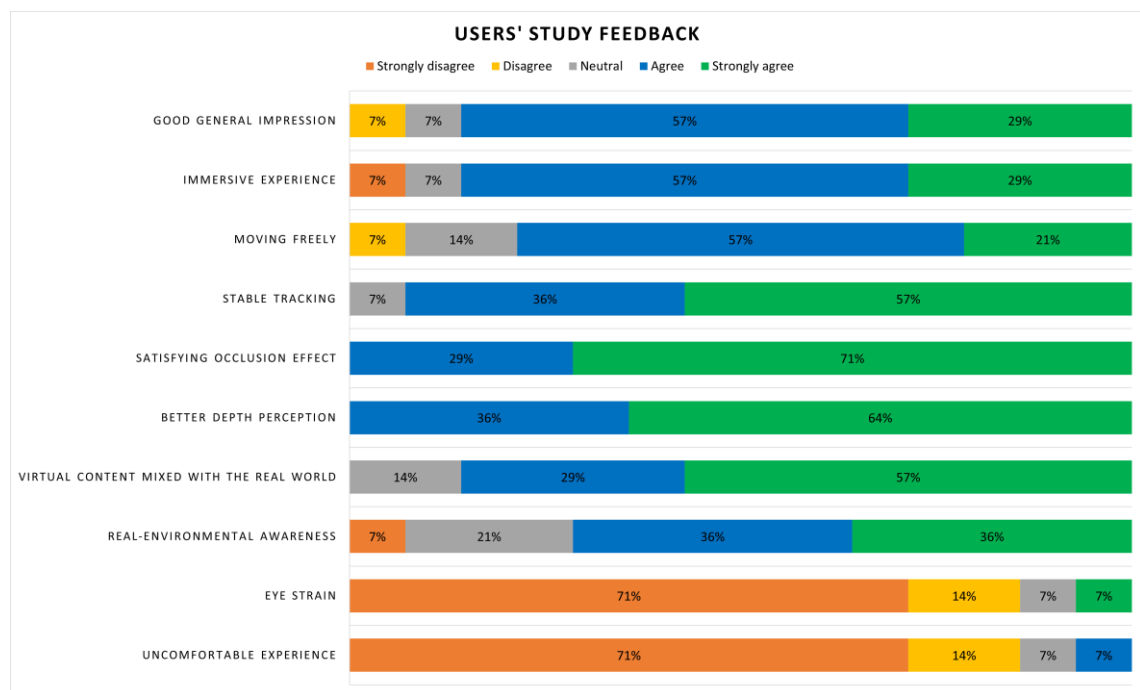


Figure 7-12: Summary of the users' study feedback.

Furthermore, the participants were asked for any further comments. One participant added that he felt a sore eye because of a health problem he already had. While others commented that the vision was not completely clear in areas directly exposed to sunlight. Regarding the realism of the design, one of the participants indicated that “objects did not look real” to give the feeling that the design is mixed with reality. As for the possibility of seeing the parked cars, a participant commented that he could not see them.

## 8 RESULTS DISCUSSION

This chapter discusses the findings from Chapter 7 and then compares them to similar findings from related research.

The results show that by using Spatial Anchors in conjunction with World Locking Tools, it is possible to anchor the augmented content in the intended location, which would then answer the research question. *How to anchor digital content to the real-world coordinate system in the study area?* The results also indicate the reliability of the visual tracking system and IMU sensors in maintaining a stable tracking of the user position throughout the usage of the application, which is almost indistinguishable from the results of [38]. The responses from the small study group also support this. By comparing different test scenarios where a different number of Spatial Anchors were used, there is no noticeable change in the maximum linear error. But the angular error tends to get greater when a smaller number of Spatial Anchors are used. The speed and robustness of localization when Spatial Anchors are used can be attributed to two reasons. The first is the amount of data used from different sensors, and the second is the advantage of using the dedicated processor for the SLAM algorithm, which is aligned with previous findings [35]. The results of employing Vuforia Area Target, on the other hand, were the opposite of what was expected, as the tracking was delayed and unreliable, which may be produced by solely using the visual camera, despite the mapping of the real-world environment was performed before ahead which lowered the required computing power. Also, the Area Target requires extra hardware to scan the environment which can be a limitation if those were missing.

The changes in environmental conditions in the study area may have an impact on the tracking process, as the tracking system relies primarily on the extraction of natural features, which may become unavailable because of changes in the scene, such as snow covering the ground or falling leaves. However, the presence of other fixed features in the area would limit this effect.

Looking at the visual effect of the developed occlusion screening method based on the in-device scanned Spatial Map, we can conclude that it was possible to create an occlusion mask by using a ToF sensing camera in real-time for small features in the environment, such as statue and trees, where the occlusion effect was sufficient, which confirm previous results [13]. Meanwhile, the results show a limited capability of using the Spatial Map for occluding moving objects, e.g., cars, due to the generated noise. This in turn constitutes an answer to the second research question: *How to use in-device scanning capability for dynamic object occlusion?* Although, the limited scan range justifies the necessity of a 3D building model.



The occlusion effect based on the 3D building model was achieved and the visual effect can be seen from different observation points based on the line-of-sight between the observer and the digital content. Also, the augmented content was fully visible when there was no building on the line-of-sight. According to these findings, the study's occlusion handler is performing as intended; it mimics the impact that would have occurred if the virtual object had been a real object placed at the same position and answer the third research question: *Is it possible to enrich the scanned and reconstructed spatial mesh map with a 3D building model to improve the outdoor occlusion effect?* However, there were some issues with aligning the 3D building model with the actual building in the real environment, particularly around the edges, which might indicate that the occlusion effect was not pixel-perfect. It was not the intention of this study to achieve a pixel-perfect occlusion effect; rather, it was to provide a sufficient occlusion effect to provide users with a sense of depth. Additionally, the findings of the small research sample suggest that the established occlusion screening effect provided them with a better comprehension of the depth, where the virtual objects seemed to be mixed with the real objects. Nevertheless, more user experiments with a larger number of participants from a diversified selection group are required to determine whether the suggested occlusion management function achieves its intended effects for an entire community of AR users.

Because the LOD1 3D building model did not include the whole number of polygons, the performance evaluation did not indicate any differences when the occlusion effect was enabled. Consequently, the FPS remained consistent and did not cause eye strain for the participants in the users' study, which was conducted after the conclusion of the trial. On the other hand, employing a more detailed building model by employing various methods, such as structure from motion, which contains more polygons, could result in poor performance because it has been discovered that there is a direct relationship between the number of polygons and the performance of the model, which is in complete agreement with the findings [16].

As previously discussed, the use of a LOD2 building model in this study will not contribute to the achievement of the study's main goal and will not improve the accuracy of the occlusion effect, because the user is viewing the scene from a ground perspective and the shape of the roofs of the buildings does not play a significant role. More importantly, manually developing building masks would require more time and effort, which may limit the application's potential to be implemented in the different study areas. Furthermore, it is not expected that enhancing the accuracy of the occlusion screening effect will improve the results in any way.

By comparing the results of the proposed solution with the previous studies, the integration of in-device scanned data and the 3D building model provided a more realistic occlusion effect, which was confirmed by the results of the users' study. All previous studies used only one

method, either 3D sensing or the use of 3D environmental representation, and did not combine the two methods. Also, there were no problems with the height of the buildings because the height of the buildings is known to be the opposite of what [1] had to deal with when they generalized the height of the OSM data. Furthermore, the proposed method does not require any additional devices and can run in real-time as well.

One of the challenges in selecting a design was the restricted computer capability of the HoloLens, which prevented the incorporation of complex designs. As a result, the potential to create more realistic designs was restricted, for example, by including more physics in the design, such as air movement simulation or change of lightening. An additional constraint was generated by the idea of the device's screen, which is a see-through screen with a limited brightness, resulting in decreased visibility in bright environments. However, in general, users' satisfaction with the application prototype was positive.

Users were able to modify various parameters and see the results of their choices, and the user menu could be used to allow the option to visualize multiple design concepts so that the user could see the differences and compare those designs. Following the gathering of input, urban planners and design makers may assess each design in terms of its acceptability by the public later in the design process. The ability to observe the planned changes will also encourage the inhabitants and enhance their willingness to embrace the redesign of the neighborhood.

## 9 CONCLUSION AND FUTURE WORKS

In summary, this study has developed an AR application to achieve an immersive and representative experience as much as possible by addressing the problems of anchoring the digital objects to their true position in the real-world environment and the occlusion problem as well, where the augmented content is hidden when located behind a real object. This application can be used in the field of urban planning to enable planners to see and experience different design options in real life. Also, the application enables the public to participate in the decision-making process, where feedback from the local community can be incorporated into the planning of future scenarios.

The in-device Spatial Mapping capability was explored for use as a real-time occlusion mask in the study area, where it was found that it works well for static objects, as it was able to scan objects and form a three-dimensional model that works as a mask. As for the dynamic objects e.g., moving cars and pedestrians, there were many noises generated by the scanning process that affected the final experience in general. Therefore, this study proposed the use of a 3D building model to act as an occlusion mask. The geometry of the building model was aligned with the real-life equivalent and rendered using a custom Shader wrapped in a material, where the augmented content was not rendered, or in other words, occluded, based on the relationship with the observer location. If the digital content was completely or partially behind a real building that was in the line of sight with the observer, then the digital content would be occluded. Activating an occlusion screen did not affect the overall performance of an application. The results of a performance evaluation did not show how the overall performance of the application changed when this effect was turned on.

Despite the absence of GPS tracking, this research provided a method to anchor and maintain the position of the digital content as well as the building model occlusion mask using a computer vision approach. The first of two possible solutions were tested. The first was using Vuforia Area Target, which required a pre-scan of the environment and didn't provide sufficient reliability of tracking. The other one is Spatial Anchors in compensation for World Locking Tools. The results of the users' study also showed that the tracking system was stable during their experience of the application.

Concerning assessing the proposed occlusion screening effect, a study was conducted on users from different backgrounds, where 14 users tested the application and evaluated the different occlusion options. Most of the participants reported that activating the occlusion option improved their depth perception and provided a more immersive experience.

This thesis although limited by time and resource availability has opened the possibility for future research including:

- Improve the in-device spatial mapping to detect and eliminate the noise caused by moving objects. Also, developing a semantic scene understanding to automatically complete missing surfaces and fill the holes in the 3D objects, as well as simplification of the reconstructed surfaces to have straight and sharp edges, especially around the building corners.
- Also, this thesis work can be expanded by incorporating a virtual assistant to guide users through the experiment and collect their feedback. And adding auxiliary sounds by exploiting the three-dimensional sound system to create a more realistic experience.
- Further intriguing future research may include using the same 3D building model for other types of physical simulations, such as shadowing, air movement models, and collisions.
- Since the field of augmented reality is changing so rapidly in terms of the hardware and SDKs used, it may be possible to create applications capable of creating a more realistic and complex experience. So, keeping the work updated with ongoing rapid developments in technology is another challenge to all the research work like this.

## REFERENCES

- [1] J. Kasper, M. P. Edwardsson, and M. Romero, "Occlusion in outdoor augmented reality using geospatial building data," in *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, Nov. 2017, vol. Part F131944, pp. 1–10, doi: 10.1145/3139131.3139159 [Online]. Available: <https://dl.acm.org/doi/10.1145/3139131.3139159>
- [2] P. Fortin and P. Hebert, "Handling Occlusions in Real-time Augmented Reality : Dealing with Movable Real and Virtual Objects," in *The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, 2006, pp. 54–54, doi: 10.1109/CRV.2006.38 [Online]. Available: <http://ieeexplore.ieee.org/document/1640409/>
- [3] A. Cirulis and K. B. Brigmanis, "3D Outdoor Augmented Reality for Architecture and Urban Planning," *Procedia Computer Science*, vol. 25, pp. 71–79, Dec. 2013, doi: 10.1016/j.procs.2013.11.009. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1877050913012143>
- [4] M. Allen, H. Regenbrecht, and M. Abbott, "Smart-phone augmented reality for public participation in urban planning," in *Proceedings of the 23rd Australian Computer-Human Interaction Conference on - OzCHI '11*, Nov. 2011, pp. 11–20, doi: 10.1145/2071536.2071538 [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2071536.2071538>
- [5] G. A. Lee, A. Dunser, Seungwon Kim, and M. Billinghurst, "CityViewAR: A mobile outdoor AR application for city visualization," in *2012 IEEE International Symposium on Mixed and Augmented Reality - Arts, Media, and Humanities (ISMAR-AMH)*, Nov. 2012, pp. 57–64, doi: 10.1109/ISMAR-AMH.2012.6483989 [Online]. Available: <http://ieeexplore.ieee.org/document/6483989/>
- [6] J. Hui, "Outdoor Mobile Augmented Reality in Urban Planning; Concepts of Visualizing Focus and Context Using Grayscale Filter in Video-based AR," *Master-Thesis Photogrammetry and Geoinformatics*, HFT Stuttgart, 2021 [Online]. Available: <https://ugl.hft-stuttgart.de/content/justin/index.html>
- [7] J. Kysela and P. Štorková, "Using Augmented Reality as a Medium for Teaching History and Tourism," *Procedia - Social and Behavioral Sciences*, vol. 174, pp.

- 926–931, Feb. 2015, doi: 10.1016/j.sbspro.2015.01.713. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1877042815007648>
- [8] “Intelligent tourism and cultural information through ubiquitous services | ITAC-ITUS Project | Fact Sheet | FP6 | CORDIS | European Commission.” [Online]. Available: <https://cordis.europa.eu/project/id/034520>. [Accessed: Jan. 06, 2022]
- [9] “Memory of Nations.” [Online]. Available: <https://www.memoryofnations.eu/en>. [Accessed: Jan. 06, 2022]
- [10] H. Dastageeri, M. Storz, and V. Coors, *SPIRIT – Videobasierte mobile Augmented Reality Lösung zur interaktiven Informationsvermittlung*. 2015.
- [11] Microsoft, “‘Hollywood Dream Machines’ brings iconic cars to life | Microsoft In Culture - YouTube,” 2019. [Online]. Available: [https://www.youtube.com/watch?v=b\\_8u5ptgcaI&t=102s&ab\\_channel=Microsoft](https://www.youtube.com/watch?v=b_8u5ptgcaI&t=102s&ab_channel=Microsoft). [Accessed: Sep. 27, 2021]
- [12] B. C. Kress, *Optical Architectures for Augmented-, Virtual-, and Mixed-Reality Headsets*. SPIE, 2020 [Online]. Available: <https://www.spiedigitallibrary.org/ebooks/PM/Optical-Architectures-for-Augmented--Virtual--and-Mixed-Reality/eISBN-9781510634343/10.1117/3.2559304>
- [13] J. Fischer, B. Huhle, and A. Schilling, “Using Time-of-Flight Range Data for Occlusion Handling in Augmented Reality,” *IPT-EGVE 2007 - 13th Eurographics Symposium on Virtual Environments, 10th Immersive Projection Technology Workshop*, pp. 109–116, Jan. 2007, doi: 10.2312/EGVE/IPT\_EGVE2007/109-116.
- [14] A. H. Behzadan and V. R. Kamat, “Scalable Algorithm for Resolving Incorrect Occlusion in Dynamic Augmented Reality Engineering Environments,” *Comput. Aided Civ. Infrastructure Eng.*, vol. 25, pp. 3–19, 2010.
- [15] J. Li and X. Fan, “Outdoor augmented reality tracking using 3D city models and game engine,” in *2014 7th International Congress on Image and Signal Processing*, Oct. 2014, pp. 104–108, doi: 10.1109/CISP.2014.7003758 [Online]. Available: <http://ieeexplore.ieee.org/document/7003758/>
- [16] O. M. Tepper *et al.*, “Mixed Reality with HoloLens,” *Plastic and Reconstructive Surgery*, vol. 140, no. 5, pp. 1066–1070, Nov. 2017, doi:

- 10.1097/PRS.0000000000003802. [Online]. Available:  
<http://journals.lww.com/00006534-201711000-00034>
- [17] A. Sharma, R. Mehtab, S. Mohan, and M. K. Mohd Shah, “Augmented reality – an important aspect of Industry 4.0,” *Industrial Robot: the international journal of robotics research and application*, vol. ahead-of-print, no. ahead-of-print, Nov. 2021, doi: 10.1108/IR-09-2021-0204. [Online]. Available: <https://www.emerald.com/insight/content/doi/10.1108/IR-09-2021-0204/full/html>
- [18] M. Filipenko, A. Angerer, A. Hoffmann, and W. Reif, “Opportunities and Limitations of Mixed Reality Holograms in Industrial Robotics,” Jan. 2020 [Online]. Available: <http://arxiv.org/abs/2001.08166>
- [19] Y. Liu, H. Dong, L. Zhang, and A. el Saddik, “Technical Evaluation of HoloLens for Multimedia: A First Look,” *IEEE MultiMedia*, vol. 25, no. 4, pp. 8–18, Oct. 2018, doi: 10.1109/MMUL.2018.2873473. [Online]. Available: <https://ieeexplore.ieee.org/document/8493267/>
- [20] “Mercedes-Benz USA: Schnellere Wartung mit Microsoft HoloLens 2 | News Center Microsoft.” [Online]. Available: <https://news.microsoft.com/de-de/mercedes-benz-usa-nutzt-hololens2-fuer-wartung/>. [Accessed: Jan. 07, 2022]
- [21] A. Farasin, F. Peciarolo, M. Grangetto, E. Gianaria, and P. Garza, “Real-time Object Detection and Tracking in Mixed Reality using Microsoft HoloLens,” in *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, Jan. 2020, pp. 165–172, doi: 10.5220/0008877901650172 [Online]. Available: <http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0008877901650172>
- [22] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, “Augmented reality: a class of displays on the reality-virtuality continuum,” in *Telemanipulator and Telepresence Technologies*, Dec. 1995, vol. 2351, pp. 282–292, doi: 10.1117/12.197321 [Online]. Available: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=981543>
- [23] T. Höllerer and S. Feiner, “Mobile Augmented Reality,” in *Telegeoinformatics: Location-Based Computing and Services*, 2004, pp. 221–260 [Online]. Available: <https://sites.cs.ucsb.edu/~holl/pubs/hollerer-2004-tandf.pdf>. [Accessed: Feb. 21, 2022]



- [24] SUTHERLAND IE, “A head-mounted three dimensional display,” vol. 33, no. pt 1, pp. 757–764, Dec. 1968, doi: 10.1145/1476589.1476686. [Online]. Available: <https://www.scinapse.io/papers/1994281102>. [Accessed: Jan. 07, 2022]
- [25] T. A. Furness, “The Super Cockpit and its Human Factors Challenges,” *Proceedings of the Human Factors Society Annual Meeting*, vol. 30, no. 1, pp. 48–52, Sep. 1986, doi: 10.1177/154193128603000112. [Online]. Available: <http://journals.sagepub.com/doi/10.1177/154193128603000112>. [Accessed: Jan. 07, 2022]
- [26] V. Coors and U. Jasnoch, “Using Wearable GIS in Outdoor Applications,” in *IMC '98. Interactive Applications of Mobile Computing. Proceedings*, 1998.
- [27] “Smartphone users 2026 | Statista.” [Online]. Available: <https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>. [Accessed: Jan. 08, 2022]
- [28] A. G. Taylor, *Develop Microsoft HoloLens Apps Now*. Berkeley, CA: Apress, 2016 [Online]. Available: <http://link.springer.com/10.1007/978-1-4842-2202-7>
- [29] D. Amin and S. Govilkar, “Comparative Study of Augmented Reality Sdk’s,” *International Journal on Computational Science & Applications*, vol. 5, no. 1, pp. 11–26, Feb. 2015, doi: 10.5121/ijcsa.2015.5102. [Online]. Available: <http://www.airccse.org/journal/ijcsa/papers/5115ijcsa02.pdf>
- [30] B. Dias, B. Keller, and S. Delabrida, “Evaluation of augmented reality SDKs for classroom teaching,” in *Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems*, Oct. 2019, pp. 1–11, doi: 10.1145/3357155.3358447 [Online]. Available: <http://dl.acm.org/doi/10.1145/3357155.3358447>
- [31] “artoolkitX · GitHub.” [Online]. Available: <https://github.com/artoolkitx>. [Accessed: Jan. 09, 2022]
- [32] P. Nowacki and M. Woda, “Capabilities of ARCore and ARKit Platforms for AR/VR Applications,” 2020, pp. 358–370 [Online]. Available: [http://link.springer.com/10.1007/978-3-030-19501-4\\_36](http://link.springer.com/10.1007/978-3-030-19501-4_36)
- [33] “ARKit | Apple Developer Documentation.” [Online]. Available: <https://developer.apple.com/documentation/arkit/>. [Accessed: Jan. 09, 2022]

- [34] “Introduction to Depth on Android | ARCore | Google Developers.” [Online]. Available: <https://developers.google.com/ar/develop/java/depth/introduction>. [Accessed: Jan. 09, 2022]
- [35] A. Cīrulis, K. Brigmanis-Brigis, and G. Zvejnieks, “Analysis of Suitable Natural Feature Computer Vision Algorithms for Augmented Reality Services,” *Baltic Journal of Modern Computing*, vol. 8, no. 1, Jan. 2020, doi: 10.22364/bjmc.2020.8.1.10. [Online]. Available: [http://www.bjmc.lu.lv/fileadmin/user\\_upload/lu\\_portal/projekti/bjmc/Contents/8\\_1\\_10\\_Cirulis.pdf](http://www.bjmc.lu.lv/fileadmin/user_upload/lu_portal/projekti/bjmc/Contents/8_1_10_Cirulis.pdf)
- [36] “Basics of AR: Anchors, Keypoints & Feature Detection – andreasjakl.com.” [Online]. Available: <https://www.andreasjakl.com/basics-of-ar-anchors-keypoints-feature-detection/>. [Accessed: Jan. 10, 2022]
- [37] “Spatial mapping - Mixed Reality | Microsoft Docs.” [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping>. [Accessed: Jan. 10, 2022]
- [38] P. Hübner, K. Clintworth, Q. Liu, M. Weinmann, and S. Wursthorn, “Evaluation of HoloLens Tracking and Depth Sensing for Indoor Mapping Applications,” *Sensors*, vol. 20, no. 4, p. 1021, Feb. 2020, doi: 10.3390/s20041021. [Online]. Available: <https://www.mdpi.com/1424-8220/20/4/1021>
- [39] C. Slattery, M. Manager, Y. Shida, and P. Marketing Manager, “ADI ToF Depth Sensing Technology: New and Emerging Applications in Industrial, Automotive Markets, and More,” 2019.
- [40] “Basics of AR: SLAM – Simultaneous Localization and Mapping – andreasjakl.com.” [Online]. Available: <https://www.andreasjakl.com/basics-of-ar-slam-simultaneous-localization-and-mapping/>. [Accessed: Jan. 11, 2022]
- [41] C. Cadena *et al.*, “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016, doi: 10.1109/TRO.2016.2624754. [Online]. Available: <http://ieeexplore.ieee.org/document/7747236/>
- [42] “Coordinate systems - Mixed Reality | Microsoft Docs.” [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/design/coordinate-systems>. [Accessed: Jan. 13, 2022]

- [43] Microsoft, “Spatial anchors - Mixed Reality | Microsoft Docs,” 2019. [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-anchors>. [Accessed: Sep. 23, 2021]
- [44] Microsoft Developer, “(364) Developing Mobile Augmented Reality (AR) Applications with Azure Spatial Anchors - BRK2034 - YouTube,” 2019. [Online]. Available: [https://www.youtube.com/watch?v=CVmfP8TaqNU&list=PLzBynCrxxFbvkMWvZzQ7tNzuaEGe4\\_Rnp&index=1&t=4s&ab\\_channel=MicrosoftDeveloper](https://www.youtube.com/watch?v=CVmfP8TaqNU&list=PLzBynCrxxFbvkMWvZzQ7tNzuaEGe4_Rnp&index=1&t=4s&ab_channel=MicrosoftDeveloper). [Accessed: Sep. 23, 2021]
- [45] Microsoft, “Coarse relocation - Azure Spatial Anchors | Microsoft Docs,” 2021. [Online]. Available: <https://docs.microsoft.com/en-us/azure/spatial-anchors/concepts/coarse-reloc>. [Accessed: Sep. 23, 2021]
- [46] “Unity - Manual: Shaders.” [Online]. Available: <https://docs.unity3d.com/Manual/Shaders.html>. [Accessed: Jan. 14, 2022]
- [47] A. Zucconi, “A gentle introduction to shaders in Unity - Shader tutorial,” Jun. 10, 2015. [Online]. Available: <https://www.alanzucconi.com/2015/06/10/a-gentle-introduction-to-shaders-in-unity3d/>. [Accessed: Jan. 14, 2022]
- [48] F. Biljecki, H. Ledoux, and J. Stoter, “An improved LOD specification for 3D building models,” *Computers, Environment and Urban Systems*, vol. 59, pp. 25–37, Sep. 2016, doi: 10.1016/j.compenvurbsys.2016.04.005. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0198971516300436>
- [49] “Unity - Manual: Unity User Manual.” [Online]. Available: <https://docs.unity3d.com/Manual/index.html>. [Accessed: Jan. 15, 2022]
- [50] D. Ungureanu *et al.*, “HoloLens 2 Research Mode as a Tool for Computer Vision Research,” Aug. 2020 [Online]. Available: <http://arxiv.org/abs/2008.11239>
- [51] Microsoft, “HoloLens 2 hardware | Microsoft Docs,” 2020. [Online]. Available: <https://docs.microsoft.com/en-us/hololens/hololens2-hardware>. [Accessed: Sep. 22, 2021]
- [52] Microsoft, “Comfort - Mixed Reality | Microsoft Docs,” 2020. [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/design/comfort>. [Accessed: Sep. 23, 2021]

- [53] “World Locking Tools for Unity Documentation.” [Online]. Available: <https://microsoft.github.io/MixedReality-WorldLockingTools-Unity/README.html>. [Accessed: Jan. 19, 2022]
- [54] B. Tag, J. Shimizu, C. Zhang, K. Kunze, N. Ohta, and K. Sugiura, “In the Eye of the Beholder: The Impact of Frame Rate on Human Eye Blink,” in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, May 2016, vol. 07-12-May-2016, pp. 2321–2327, doi: 10.1145/2851581.2892449 [Online]. Available: <https://dl.acm.org/doi/10.1145/2851581.2892449>
- [55] Microsoft, “Hologram stability - Mixed Reality | Microsoft Docs,” 2020. [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/develop/platform-capabilities-and-apis/hologram-stability>. [Accessed: Sep. 23, 2021]
- [56] R. Vassallo, A. Rankin, E. C. S. Chen, and T. M. Peters, “Hologram stability evaluation for Microsoft HoloLens,” in *Medical Imaging 2017: Image Perception, Observer Performance, and Technology Assessment*, Mar. 2017, vol. 10136, p. 1013614, doi: 10.1117/12.2255831 [Online]. Available: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2255831>
- [57] R. Likert, “A technique for the measurement of attitudes,” *Archives of Psychology*, vol. 22 140, p. 55, 1932 [Online]. Available: [https://legacy.voteview.com/pdf/Likert\\_1932.pdf](https://legacy.voteview.com/pdf/Likert_1932.pdf). [Accessed: Jan. 26, 2022]

## ANNEXES

### I. Fine-Tuning Alignment Dashboard Script

```
// Shift in X-direction
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Microsoft.MixedReality.Toolkit.UI;
5
6 namespace Microsoft.MixedReality.Toolkit.Examples.Demos
7 {
8     public class xPositionSlider : MonoBehaviour
9     {
10         [SerializeField]
11         private Transform transformDesign = null;
12
13         public void OnSliderUpdated(SliderEventData eventData)
14         {
15             if (transformDesign != null)
16             {
17                 // Move the target object using Slider's eventData.NewValue
18                 transformDesign.position = new Vector3(0.92f +
eventData.NewValue, transformDesign.position.y, transformDesign.position.z);
19             }
20         }
21     }
22}

// Shift in Y-direction
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Microsoft.MixedReality.Toolkit.UI;
5
6 namespace Microsoft.MixedReality.Toolkit.Examples.Demos
7 {
8     public class yPositionSlider : MonoBehaviour
9     {
10         [SerializeField]
11         private Transform transformDesign = null;
12
13         public void OnSliderUpdated(SliderEventData eventData)
14         {
15             if (transformDesign != null)
16             {
17                 // Move the target object using Slider's eventData.NewValue
18                 transformDesign.position = new Vector3(transformDesign.po-
sition.x, -3.2f + eventData.NewValue, transformDesign.position.z);
19             }
20         }
21     }
22}
```

```
// Shift in Z-direction
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Microsoft.MixedReality.Toolkit.UI;
5
6 namespace Microsoft.MixedReality.Toolkit.Examples.Demos
7 {
8     public class zPositionSlider : MonoBehaviour
9     {
10         [SerializeField]
11         private Transform transformDesign = null;
12
13         public void OnSliderUpdated(SliderEventData eventData)
14         {
15             if (transformDesign != null)
16             {
17                 // Move the target object using Slider's eventData.NewValue
18                 transformDesign.position = new
Vector3(transformDesign.position.x, transformDesign.position.y, 3.31f +
eventData.NewValue);
19             }
20         }
21     }
22 }

// Rotate around Y-Axis

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Microsoft.MixedReality.Toolkit.UI;
5
6 namespace Microsoft.MixedReality.Toolkit.Examples.Demos
7 {
8     public class yAngleSlider : MonoBehaviour
9     {
10         [SerializeField]
11         private Transform transformDesign = null;
12
13         public void OnSliderUpdated(SliderEventData eventData)
14         {
15             if (transformDesign != null)
16             {
17                 var rotationVector = transform.rotation.eulerAngles;
18                 rotationVector.y = -119.04f + 15.0f * eventData.NewValue;
19                 //this number is the degree of rotation around Y Axis
20                 transform.rotation = Quaternion.Euler(rotationVector);
21             }
22         }
23 }
```



## II. Occlusion Shader Script

```
1 Shader "VR/SpatialMapping/Occlusion"
2 {
3     SubShader
4     {
5         // Render the Occlusion shader before all
6         // opaque geometry to prime the depth buffer.
7         Tags { "Queue"="Geometry-1" }
8
9         ZWrite On
10        ZTest LEqual
11        ColorMask 0
12
13        Pass
14        {
15            CGPROGRAM
16            #pragma vertex vert
17            #pragma fragment frag
18
19            #include "UnityCG.cginc"
20
21            struct appdata
22            {
23                float4 vertex : POSITION;
24                UNITY_VERTEX_INPUT_INSTANCE_ID
25            };
26
27            struct v2f
28            {
29                float4 position : SV_POSITION;
30                UNITY_VERTEX_OUTPUT_STEREO
31            };
32
33            v2f vert (appdata input)
34            {
35                v2f output;
36                UNITY_SETUP_INSTANCE_ID(input);
37                UNITY_INITIALIZE_VERTEX_OUTPUT_STEREO(output);
38
39                output.position = UnityObjectToClipPos(input.vertex);
40                return output;
41            }
42
43            fixed4 frag (v2f input) : SV_Target
44            {
45                return fixed4(0.0, 0.0, 0.0, 0.0);
46            }
47            ENDCG
48        }
49    }
50}
```

### III. User Menu Script

```
// Controlling Augmented Content
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class sceneObjects : MonoBehaviour
6 {
7     public GameObject model;
8
9     // Start is called before the first frame update
10    void Start()
11    {
12        model.SetActive(true);
13    }
14
15    public void showModel()
16    {
17        model.SetActive(true);
18    }
19
20    public void hideModel()
21    {
22        model.SetActive(false);
23    }
24
25}
```

```
// Spatial Map Behavior
1 using Microsoft.MixedReality.Toolkit;
2 using Microsoft.MixedReality.Toolkit.SpatialAwareness;
3 using System.Collections;
4 using System.Collections.Generic;
5 using UnityEngine;
6
7 public class scriptMesh : MonoBehaviour
8 {
9     public void meshHide()
10    {
11        // Get the first Mesh Observer available
12        var observer =
CoreServices.GetSpatialAwarenessSystemDataProvider<IMixedRealitySpatialAwaren
essMeshObserver>();
13
14        // Set to not visible
15        observer.DisplayOption = SpatialAwarenessMeshDisplayOptions.None;
16    }
17
18    public void meeshVisible()
19    {
20        // Get the first Mesh Observer available
21        var observer =
CoreServices.GetSpatialAwarenessSystemDataProvider<IMixedRealitySpatialAwaren
essMeshObserver>();
22
23        // Set to visible
24        observer.DisplayOption =
SpatialAwarenessMeshDisplayOptions.Visible;
25    }
26
27    public void meshOcclusion()
28    {
29        // Get the first Mesh Observer available
30        var observer =
CoreServices.GetSpatialAwarenessSystemDataProvider<IMixedRealitySpatialAwaren
essMeshObserver>();
31
32        // Set to visible and the Occlusion material
33        observer.DisplayOption =
SpatialAwarenessMeshDisplayOptions.Occlusion;
34    }
35
36}
```

```
// 3D Building Model Occlusion Mask Behavior
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class buBehaviour : MonoBehaviour
6 {
7     public GameObject modelVis;
8     public GameObject modelOcc;
9
10    // Start is called before the first frame update
11    void Start()
12    {
13        modelVis.SetActive(false);
14        modelOcc.SetActive(false);
15    }
16
17    public void visible()
18    {
19        modelVis.SetActive(true);
20        modelOcc.SetActive(true);
21    }
22
23    public void hide()
24    {
25        modelVis.SetActive(false);
26        modelOcc.SetActive(false);
27    }
28    public void occ()
29    {
30        modelVis.SetActive(false);
31        modelOcc.SetActive(true);
32    }
33}
```