FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Ilyas Hamadouche

AUGMENTED REALITY X-RAY VISION ON OPTICAL SEE-THROUGH HEAD-MOUNTED DISPLAYS

Master's Thesis
Degree Programme in Computer Science and Engineering
June 2018

Hamadouche I. (2016) Augmented reality X-ray vision on optical see-through head-mounted displays. University of Oulu, Degree Programme in Computer Science and Engineering. Master's thesis, 62 p.

ABSTRACT

In this thesis, we present the development and evaluation of an augmented reality X-ray system on optical see-through head-mounted displays. Augmented reality X-ray vision allows users to see through solid surfaces such as walls and facades, by augmenting the real view with virtual images representing the hidden objects. Our system is developed based on the optical see-through mixed reality headset Microsoft Hololens. We have developed an X-ray cutout algorithm that uses the geometric data of the environment and enables seeing through surfaces. We have developed four different visualizations as well based on the algorithm. The first visualization renders simply the X-ray cutout without displaying any information about the occluding surface. The other three visualizations display features extracted from the occluder surface to help the user to get better depth perception of the virtual objects. We have used Sobel edge detection to extract the information. The three visualizations differ in the way to render the extracted features. A subjective experiment is conducted to test and evaluate the visualizations and to compare them with each other. The experiment consists of two parts; depth estimation task and a questionnaire. Both the experiment and its results are presented in the thesis.

Keywords: Augmented Reality, Mixed Reality, X-Ray vision, Optical See-Through, Hololens.

TABLE OF CONTENTS

FOREWORD

This master's thesis was written in the Center for Machine Vision and Signal Analysis (CMVS) at the University of Oulu. I would like to thank my supervisor, Prof. Janne Heikkilä for his support and guidance during my studies and thesis. I would like to thank as well Prof. Christian Sandor, Prof. Alexander Plopski, and Prof. Takafumi Taketomi. This work is dedicated to my father Kamel, my grandfather Boukhatem, and grandfather Bouchakour.

Oulu, Finland June 1, 2018

Ilyas Hamadouche

ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
AV	Augmented Virtuality
CAD	Computer-Aided Design
DGPS	Differential Global Positioning System
DOE	Diffractive Optical Element
DoF	Degree of Freedom
CG	Computer Graphics
DOF	Degree of Freedom
IMU	Inertial Measurement Unit
IPD	Interpupillary distance
GB	Gigabytes
GPS	Global Positioning System
GPU	Graphics Processing Unit
LCD	Liquid Crystal Display
LCoS	Liquid Crystal on Silicon
LED	Light-Emitting Diode
FOV	Field of view
HD	High Definition
HMD	Head Mounted Display
HMPD	Head-Mounted Projector Display
HOE	Holographic Optical Element
HPC	Handy Personal Computer
Hz	Hertz
MR	Mixed Reality
ODG	Osterhout Design Group
OLED	Organic Light-Emitting Diode
OST	Optical See-Through
QR	Quick Response
RAM	Random Access Memory
RGB	Red Green Blue
RGB-D	Red Green Blue Depth
SDK	Software Development Kit
SfM	Structure from Motion
SIFT	Scale-Invariant Feature Transform
Si-OLED	Silicon-based Organic Light-Emitting Diode
SLAM	Simultaneous Tracking and Mapping
TIR	Total Internal Reflection
vSLAM	Visual Simultaneous Tracking and Mapping
VST	Video See-Through
WLAN	Wireless Local Area Network

1. INTRODUCTION

Augmented Reality (AR) is the technology of augmenting real-world environments with computer-generated images. The virtual images are rendered to appear as they belong to the real scene, but in fact, they are purely digital. AR allows bringing virtual objects to real-world environments, taking advantages of the recent advancements in both hardware and software. One of the most successful stories of AR is Pokémon Go (see Figure 1). The location-based AR mobile game gained an insane popularity among children and adults. The app allows users to locate, collect, train and battle Pokémons. These Computer Graphics (CG) creatures, known as Pokémons, appear on the screen as they are really located in the real-world environment. Since its release in July 2016, the number of users grew exponentially in a very short time, as well as revenues. Such kind of success gives remarkable attention to developers, to consider AR as a technology of the future. In academia, the story of AR started as early as in the 60s. Since the late 90s, AR technology started to see major advancements. To boost the improvements further, the research community established new scientific conferences and journals to discuss the topics related to AR technology. These topics include designing new AR headsets, improving AR tracking and registration, exploring new rendering approaches, and more.

Figure 1. Pokémon GO is a very popular AR game. Image credits: caffinegeeks.com.

X-ray visualization is one of the famous applications of AR. It allows users to see through walls and surfaces, by augmenting the real view with virtual images representing the hidden objects. This resembles the superhuman ability to see through walls and buildings in science fiction movies. The thesis presents the development of a new algorithm for AR X-ray vision. We have used Microsoft's mixed reality headset, the Hololens, as a development platform. Based on the developed algorithm, we derived four different X-ray visualizations. We have conducted a subjective study to test and evaluate the visualizations and to compare them with each other. The experiment was performed by voluntary participants and consists of two parts; depth estimation task and a questionnaire. The results of the evaluation are presented in the thesis as well.

The rest of the thesis is organized as follows. Chapter 2 explains the general definition of Augmented Reality, including key concepts and technologies related to it, like displays, tracking, registration, calibration, etc. Chapter 3 reviews optical see-through head-mounted displays. First, we cover the technology behind these displays, including optical combiners. Next, we explain the differences between optical and video see-through head-mounted displays. Then, we review the commercial optical see-through headsets available on the market. Finally, we discuss the problem of latency and the methods used to compensate for it on optical see-through displays. Chapter 4 highlights the state-of-the-art methods used in AR X-ray vision, focusing on the systems presented for optical see-through devices. Chapter 5 describes the implementation of the proposed algorithm as well as the four X-ray visualizations. Chapter 6 presents the experiment design to evaluate the proposed visualizations and the results obtained. Chapters 7 contains discussion and Chapter 8 draws conclusions of the thesis.

2. AUGMENTED REALITY

Augmented Reality (AR) aims at merging virtual objects into the view of the real scene, in a way that the virtual objects appear to be part of the real environment. This chapter is dedicated to explaining the engineering and techniques related to AR technology that enables this merging. Section 2.1 gives a detailed definition of AR and lists the systems that are inside its scope. Section 2.2 presents AR display devices, focusing on the most common displays. Section 2.3 describes some of the key terms associated with AR like occlusion and depth of field. Section 2.4 looks at the different tracking techniques used in AR. Tracking the user's position, orientation, and motion is the key component in every AR system. The last section discusses the registration of the virtual world with respect to the real world, and the calibration methods used to calibrate different tracking sensors.

2.1. Definition

The well-known definition of Augmented Reality among scholars is the one given by Azuma [1]. He defines an AR system to have the following features:

- Combines real and virtual objects in a real environment
- Runs interactively, and in real time
- Registers real and virtual objects with each other.

According to this definition, AR is not limited to visual media, or to a specific output display. Therefore, audio, haptics, and gustatory AR are included in the context. Furthermore, movies that contain computer graphics added to real environments, are not considered as AR, because they lack interactivity. Interactivity, the second characteristic listed by Azuma, requires that the user within an AR system can freely navigate the scene, and control the AR experience related to it, in real time [2]. To achieve this, the system needs to measure the user's viewpoint, and update the output according to the registration between the real and virtual worlds. Milgram [3] considered Augmented Reality as a part of the general area of Mixed Reality (MR). He defined a continuum of real-to-virtual environments (Figure 2). In AR, virtual objects are added to the real scene, while in Augmented Virtuality (AV) real objects are added to the virtual scene. The continuum gives a clear picture of the intersection between the reality and the virtuality. Any MR system that is closer to the reality more than the virtuality is considered as AR. On the other hand, anything closer to the virtuality is considered as AV.

2.2. Displays

As augmented reality merges virtual objects into real environments, AR displays should be able to display both the 3D virtual images and the real scene. The most common displays, used to achieve this, are see-through Head Mounted Displays (HMDs).

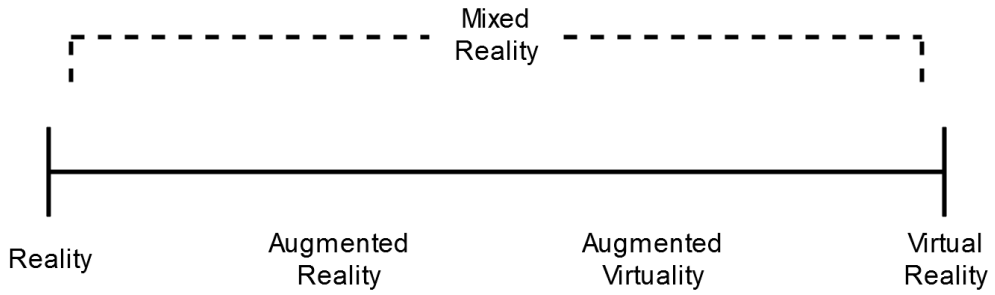

Figure 2. Milgram's continuum of Mixed Reality.

HMDs are categorized into two main types: video, and optical see-through. In the optical see-through (OST) HMDs, the real and virtual images are combined using half transparent and half reflective mirrors. The mirrors are usually placed in front of the eyes, while the imaging device is located just above them, or on the side of the display so that it does not block the view for the eyes [4]. The real world is seen as it is through the transparent part, without any modification or alternation. The virtual images are optically combined with the real world images using the reflective part of the mirrors. The optical see-through HMDs are generally simple and lightweight. The video see-through (VST) HMDs block the natural view from the eyes and captures the real world using a camera instead. The captured images are then combined with the virtual images electronically [4]. The final result is displayed to the user using a monitor placed in front of the eyes. Since the real scene is represented as digital images, several image processing algorithms can be applied to process the scene. In fact, there are less commercial video see-through HMDs in the market, than optical see-through HMDs. Usually, research groups build their own video see-through displays to experiment and develop applications related to them. Optical and video see-through HMDs are discussed in more detail in the next chapter. Despite the popularity of the see-through HMDs as AR displays, there exist other solutions including hand-held devices, hybrid optical/video see-through, spatial AR, and immersive displays. The hand-held devices, such as smartphones and tablets, are able to provide both virtual and real scenes in one view. Since these mobile devices are available and cheaper than HMDs, they are considered as the most used AR platform [2]. Next, we address briefly the different types of AR displays other than optical and video HMDs.

Hand-held/mobile displays are based on smartphones and tablet computers. They use the built-in camera to capture the real world, then merge the real and virtual images into a single view, and display it on the screen. Since the system uses the camera, many consider them as video see-through displays [5]. Figure 3 shows an example of a hand-held AR. Tracking the device with respect to the world is done mainly using the camera, yet it is also possible to use other sensors as well [2]. In some recently proposed systems, the built-in front camera is used to track the user's pose relative to the device. Such a system requires two different tracking processes, one for user-to-device, and one for device-to-world tracking [6] [7].

Hybrid optical/video see through displays are designed to use optical see-through technology for visualization, and video camera for tracking and localizing objects in the real world [8]. The real world view is kept intact, while a camera senses the en-

Figure 3. Handheld devices are cheap devices to experience AR. Credit: http://magicway.org/services/mobile-app-development-warsaw/.

vironment and helps in generating the virtual graphics, as in VST. This is very useful in applications that require the natural view of the real scene as well as the camera for image processing [9]. **Projection-based AR**, also known as spatial AR, and spatial projection, is another way to augment the real scene, but without a physical screen. The augmentation is projected, using a light projector, onto the real world geometry, as shown in Figure 4. This system does not require a physical screen, nor an optical combiner, and is perfect for public events and spatial effects on buildings and large structures [2]. Another setup proposed by Kijima and Ojika [10] is to attach a projector to a headset, instead of placing it in the environment. Such headsets are called Head-Mounted Projector Displays (HMPDs).

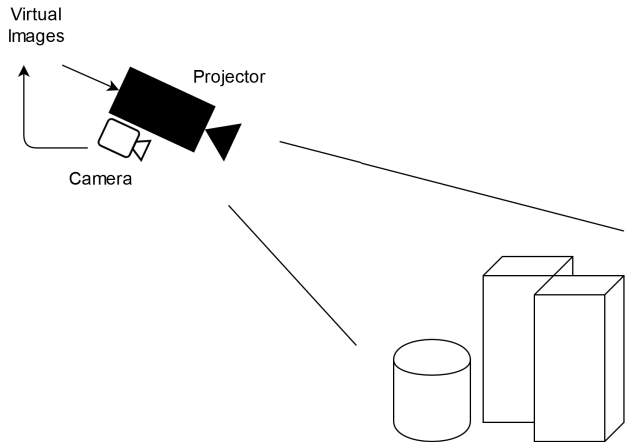

Figure 4. Projective-based AR conceptual diagram.

Fully immersive headsets totally occlude the user's view of the real world [5]. They use stereoscopic displays together with specialized lenses to deliver a 3D view to the user. Two slightly different images are served to the eyes, to help the user to sense the depth using parallax. The display is very close to the eyes and covers their entire

Figure 5. Fully immersive displays. From left to right: HTC Vive, Oculus Rift, and Sony Playstation VR. Credit: https://geekculture.co/htc-vive-oculus-rift-ps-vr-compared-who-wins-your-money/

field of view. The lenses are placed between the eyes and the display. They allow the eyes to focus on the images on the display. As the display is very close to the eyes, the images would be blurry without the lenses [11]. Motion tracking sensors are used to track head's position and orientation. The images are updated according to the tracking. This setup makes the user experience the visual presence in the non-physical world. This type of technology is suits best for Virtual Reality (VR) and AV. The system has several interaction methods to interact with the content. The input methods include hand gestures, voice commands, keyboards, game controllers, and more. There are plenty of commercial immersive displays in the market, available for the general audience. The most common headsets are Oculus Rift, Sony PlayStation VR, and HTC Vive (see Figure 5). Fully immersive headsets can be used as video see-through displays if they have one or two cameras attached in front of them. In case they do not have any, one can attach a camera manually [4].

2.3. Basic Concepts

The first attempt to create an AR device that augments the real world with digital graphics happened as early as in the 1960s. Since that, AR has seen many improvements and advances thanks to the research community. Nowadays, there are many terms associated with AR technology. Understanding them helps to acknowledge AR and its dependencies better. For example, understanding the ocularity gives an idea why some AR headsets have only one display, such as Google Glass [12], while other headsets have two displays, such as Epson BT-300[13]. In this section, we go through some of the key terms, including ocularity, field of view, depth of field, interpupillary distance, latency, and occlusion.

2.3.1. Ocularity

Ocularity refers to the mechanisms how the AR system shows images to one or two eyes. Head-mounted displays are divided into three types based on their ocularity: monocular, biocular, and binocular. A **Monocular** display serves only one eye. A small display is placed in front of an eye, while the second observes the real world freely. Such kind of low weight devices are perfect for displaying information content, but suffer from lacking stereo depth cues. **Biocular** displays serve both eyes from a single image source. These devices are mainly used for cinematic viewing and are common to be less complex and lightweight than their binocular counterparts. Yet, they also suffer from missing stereo depth cues. **Binocular** displays serves both eyes with separate image sources to provide a stereoscopic view. That is very friendly for the human vision system. The main advantages of binocular displays are that they have a large field of view, and provide stereo images, and depth cues. On the other hand, they are complex, expensive, and require more computation power compared to the other types [5].

2.3.2. Field of View

Field of view (FOV) in AR refers to the total amount of information displayed to both eyes, measured as angles in degrees. It is considered as an important feature of AR headsets [14]. The FOV of an HMD can be classified into three regions; aided (or overlay), peripheral, and occluded. Aided FOV is the most important and critical region. It tells how much the virtual images are overlayed onto the user's view. Generally, narrow FOV HMDs, whose FOV is horizontally less than $60°$, have 100% overlap. However, wide FOV HMDs (horizontal FOV more than $80°$) commonly have smaller overlap like 60%, for example [4]. Occluded FOV is the region that is blocked by the headset, in which the user cannot see the real world, nor the virtual images. The occluded region is recommended to be as small as possible. Peripheral FOV refers to the part between aided and occluded regions, where the user can see only the real world. Usually, the optical see-through HMDs have a large peripheral FOV. On the other hand, it is very hard to achieve wide overlay FOVs on OST devices, due to the design of the optical combiners. Increasing the FOV also brings more distortions and aberrations [4]. Video see-through and immersive HMDs have wide aided FOV and no or small peripheral FOV. Some applications require large FOV, while other applications, where only a few augmentations, are added to the real world view are satisfied with small FOV [8].

2.3.3. Depth of Field

The range of distances from the eye in which an object appears in focus is known as the depth of field. That is defined also as a distance between the closest and the furthest objects that the eyes can see. The human visual system can focus on a particular distance each time. It automatically adjusts the eye's accommodation to focus on a single plane where the object lays on. This plane is called the plane of focus. Objects

that lay at different depths than the focused plane are seen blurred [15]. In the case of AR, HMDs provide both the real and virtual worlds. Stereoscopic displays in general, display synthetic images of virtual objects at a fixed distance, known as fixed focal depth [2]. With optical see-through HMDs, if the real objects and the synthetic images are not at the same distance or close enough, it is impossible for the human eyes to focus on both of them at the same time. This problem is very common with transparent HMDs and known as the accommodation-vergence conflict. In the video see-through HMDs, which are not transparent, this problem does not occur, because the real objects can be defocused due to the camera. Moreover, in order to avoid blurred video images, small aperture autofocus cameras are used [4]. Adjusting the focus in HMDs can be done by manually moving the lenses towards or away from the display until the CG appears sharp to the user [16]. Several methods have been proposed to solve the accommodation-vergence conflict with automatic refocus in transparent HMDs [17] [18].

2.3.4. Interpupillary Distance

Interpupillary distance (IPD) is the distance between the centers of the pupils of the eyes, measured in millimeters. It is a key parameter for all binocular systems including stereoscopic head-mounted displays. All head-mounted displays need this parameter to accurately render the synthetic images. It helps to align the virtual camera output correctly on the display screen that is placed in front of the eyes. The distance varies between people. The average IPD for adult humans is 64 mm with the range of 50 to 70 mm. Some exeptional IPDs are within range of 45 to 80 mm [19]. Errors regarding IPD settings can cause image distortion on the display, eye strain, and headaches [5]. Most of the HMDs offer a way to adjust the IPD parameter for a proper output rendering. Some displays, like HTC Vive and Oculus Rift, provide a mechanical tool to manually adjust the IPD setting by moving the lenses. Another used strategy is to keep the lenses fixed and change the IPD using a software. All the synthetic images will be rendered according to the set value. Sony PlayStation VR and Microsoft's Hololens displays, for examples, adopt this technique [5].

2.3.5. Latency

In AR systems, latency refers to the time delay from the change of the user's pose to the moment when the display content is updated to correspond the new pose. It is a very important parameter and usually measured in milliseconds. The latency reflects the total time the whole system takes to complete one full loop from sensing to displaying. This loop includes tracking, processing the results, rendering, and then updating the display output. If the latency is high, inconsistency between visual and vestibular systems happens and user observes registration errors and disorientation and starts suffering from motion sickness [4]. For video see-through HMDs, the issue of high latency is not as critical as in the optical see-through HMDs. The latency is minimized by displaying the captured image of the real scene until the synthetic image is fully rendered. This eliminates the time lag between the real and virtual worlds, but

introduce a small delay to the real scene. Generally, for a good AR experience using HMDs, the latency should be less than 60ms [20]. Several algorithms have been proposed to compensate, minimize or eliminate the latency. Time warping and prediction methods such as extended Kalman filter (EKF) are used to minimize the lag [21] [22]. The latency is discussed in the next chapter in more detail.

2.3.6. Occlusion

Occlusion is a necessary ability for all AR systems. It plays a major role in increasing the depth perception and the overall realism of the system. In fact, it is considered as an important depth cue, where from the other depth cues depend on [23] [24]. The human visual system uses the overlappings between objects to understand the structure of the seen 3D scene. Any occlusion errors confuse the visual system. In HMDs, preserving the depth order is important for realistic rendering. To achieve this, the occlusions should be handled correctly. If all the objects are real, then they are naturally ordered. The human visual system automatically senses the depth order. If the objects are virtual, the z-buffer technique is used to order them before sending the final image to the display. However, if the scene contains both real and virtual objects, handling occlusions becomes challenging. In video see-through HMDs, it is relatively easy, because both the real and virtual worlds are represented by images. If the geometric structure of the real scene is known, the z-buffer can be used as well to place the virtual objects in front of the real objects and vice versa [8]. The geometric order can be obtained using a cheap RGB-D camera, for example [4]. On the other hand, due to the display capabilities, the occlusion handling occlusion is challenging for optical see-through HMDs, even if the depth order of the real scene is given. As the real world is seen through half-transparent mirrors, it is hard, or even impossible, to stop the natural light to reach the eyes. Thus, making the virtual objects to look like they are in front of the real objects is difficult. As a solution, Kiyokawa et al. [25] proposed a system where a liquid crystal display (LCD) is placed in front of the mirrors, to block the light coming from the real scene when needed. The proposed concept is illustrated in Figure 6.

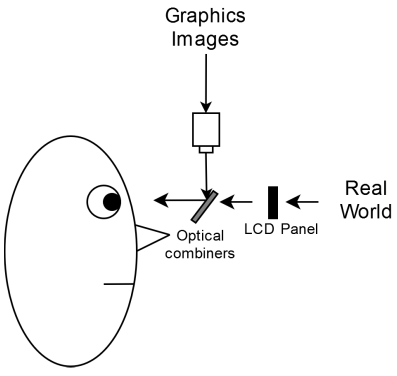

Figure 6. An LCD panel for optical see-through devices to help handling the occlusions.

2.4. Tracking

Tracking is the key process in all AR systems. It refers to the continuous measurement of user's position, orientation, and motion in the 3D space. This process is necessary for AR and VR systems to physically locate the user's head with respect to the 3D environment. According to the tracking results, the view content is updated at the display output to match the user's position and orientation. Other entities, such as cameras, displays and input devices, can be tracked as well. There are many approaches to track objects in the current AR systems. In this section, we explore sensor-based tracking, optical tracking, sensor fusion, and simultaneous tracking and mapping (SLAM).

2.4.1. Sensor-based tracking

The rapid development of smartphones and tablets made a variety range of sensors cheap and available. Their output is not as accurate as their expensive and professional counterparts, but they are enough for performing several kinds of tracking. To list a few, methods and devices like GPS, magnetometers, gyroscopes, and linear accelerometers, are commonly used for AR tracking. **Global Positioning System** (GPS) measure the position using signals emitted from the satellites in Earth's orbit. For obtaining good accuracy, four or more satellites are needed. However, such signals are very sensitive to external disturbance and noise. Buildings and grand surfaces may weaken or block them. Solutions like differential GPS (DGPS) achieves higher accuracy. DGPS uses correction signals, received from the ground stations which their positions are known in advance. GPS technology is not suitable for precise real-time tracking that is required for AR registration [2].

A **magnetometer**, or electronic compass, uses Earth's magnetic field to determine the direction with respect to the magnetic north. The results are usually given along three axes. A **Gyroscope** is an inertial sensor that measures rotational velocity. To get a 3 DOF measurement, a setup of three orthogonal gyroscopes is used. **Linear accelerometers** estimate accelerations using the piezoresistive effect and/or sensing the change of internal electric capacity caused by the movements. It is very important to notice that every sensor has its own advantages as well as disadvantages. Furthermore, in general, the discussed sensors suffer from lacking high accuracy measurements, which are needed for high-quality AR registration [2]. As a result, in real time tracking systems, a single sensor is rarely used alone.

2.4.2. Optical tracking

Optical tracking refers to the use of cameras to track movements. There are a variety of optical tracking systems being used, but all of them use a single or multiple cameras as a source of data to measure the pose. Since cameras are cheap and provide rich easy-to-process data of the environment, they are widely used in AR tracking. Both camera technology and computer vision algorithms are under extensive academic and industrial research. It is expected that they will see many improvements in the near future [2]. Optical tracking systems can be classified into two categories based on

the targets they track. The targets can be either manually created markers or natural features [4]. Markers, also known as fiducial markers, contain usually a predefined pattern or features. They are placed on specific positions in a particular order in the 3D space. When the system starts, one or multiple cameras look for them in the real space. Whenever a marker is seen on the images, the position and orientation of the object or the tracking camera can be estimated accordingly. If the markers are not used, the system tries to extract natural features from the scene using computer vision algorithms. Such features are known as **interest points** or **keypoints**. Natural feature tracking requires higher image resolution and more computational power. Moreover, the features should be easy to find and remain in fixed positions [2].

2.4.3. Sensor Fusion

Building a sophisticated tracking system requires the use of all possible data from the available sensors. When aiming at increasing the performance and the quality of tracking, different sensor technologies are used simultaneously. However, this setup increases the complexity, cost, weight, power consumption, and require more calibration procedures [2]. The combination of multiple sensors to improve the quality of the measurement is known as sensor fusion. In AR tracking, many good results have been achieved using the sensor fusion [26] [27] [28]. According to [29], the sensor fusion systems are classified into three main categories based on their configuration. In **complementary sensor fusion**, sensors are not dependent on each other. Every sensor measures a single degree of freedom and the results are combined at the end. **Competitive sensor fusion** occurs when every sensor measures the same degree of freedom. The final measurement is a combination of the individual measurements calculated with a mathematical model. This configuration is robust against uncertain and erroneous measurements [30]. In **cooperative sensor fusion**, a given sensor uses the data measured by other sensors to obtain its measurements. This happens when a property cannot be measured with a single sensor.

2.4.4. SLAM

Simultaneous Localization and Mapping (SLAM) was first proposed by John J. Leonard and Hugh Durrant-Whyte [31]. SLAM is a technique used in robotics to help mobile robots to navigate in unknown environments. It creates a map of the surrounding area around the robot, tracks the position of the robot with respect to the map and uses it for navigation as well. All this happens at the same time. SLAM has seen many improvements. It can run in real-time and use different sensors for tracking, including IMU and RGB-D sensors. If SLAM is based only on visual data and uses only cameras, then it is called visual SLAM (vSLAM) [32]. Today, SLAM and vSLAM are being used in many applications besides robotics, including AR systems. For example, they are used as key components in Google Tango AR computing platform [33]. Tango devices, including Lenovo Phab 2 Pro and Asus ZenFone AR, are able to navigate smoothly, measure the space of a real object, provide 3D data of the environment and,

in general, provide a good AR experience. Unfortunately, Google has decided not to support Tango anymore and prefer ARCore instead [34].

The approach proposed in [35], called Parallel Tracking and Mapping (PTAM), separates the tracking from the mapping by creating two parallel threads for each. PTAM executes the two threads in parallel and allows them to communicate with each other (see Figure 7). The tracking thread uses the features stored in the map. Then it seeks their correspondences in every frame. Based on the obtained correspondences, the user's pose is estimated. If new strong features are detected in the current tracking frame and pass a minimum requirement, they are then added to the map. Usually, the tracking thread runs much faster than the mapping. The tracking runs at the full frame rate of the system, e.g. 30 or 60 Hz, while the mapping runs once per one or few seconds [2].

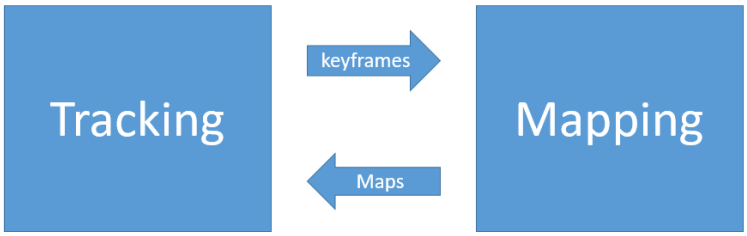

Figure 7. In PTAM, the tracking and the mapping are executed on parallel.

2.5. Registration and Calibration

In AR, the registration means the alignment of the coordinate systems of the virtual and real scenes [36]. It is extremely important for see-through devices, to display virtual objects in their correct positions, and keep them registered with respect to the real world. High-quality tracking is required for a good registration and a good AR experience. There are two types of registration: static and dynamic. Usually, the static registration happens at the beginning of the AR system when the user is not moving. A common global coordinate system is then established between the real and virtual scenes. This ensures that both the real and virtual worlds use the same coordinate system. The dynamic registration is done when the user is moving and needs continuous tracking [2]. It maintains the link between the real and virtual worlds when the user's pose changes.

A calibration is another important process in all HMDs. It is defined as a process of comparing measurements of one device with measurements of a reference device. The measurements of the reference device are considered as the ground truth. The parameters of the device to be calibrated are tuned until the two measurements are equal. In AR, the tracking sensors need to be calibrated at least once. The calibration improves the measurement accuracy. Some sensors require the calibration each time the operating system starts, others require it continuously. Camera calibration is one of the important processes. The camera calibration defines the internal camera parameters which are used to compensate for the lens distortion, for example [2].

3. OPTICAL SEE-THROUGH HEAD MOUNTED DISPLAYS

The objective of this thesis is to implement an augmented reality X-ray vision on optical see-through HMDs. In this section, we discuss the optical see-through displays, whereas X-ray vision and its related work, are discussed in the next chapter. Highlighting the advantages of OST-HMDs, as well as their disadvantages, helps to understand the challenges related to development of an X-ray system for such displays. We start this chapter by presenting the concepts and technologies enabling OST-HMDs, reviewing the different approaches used by companies when building their OST headsets. Next, we compare the optical and video see-through displays. We will see that development of an X-ray vision system for VST is easier compared to OST. This explains why the most of the related work of X-ray vision has been presented for VST displays, while very few works have been proposed for OST. Then, we explore the problem of latency in HMDs, and the techniques used to compensate for it. The system latency is a very critical problem for OST in particularly, and for all HMDs generally. Finally, we present a brief overview of the available OST headset in the market.

3.1. Optical See-Through Displays

Optical see-through head-mounted displays are one type of augmented reality headsets that mixes real and virtual images in a single view. They use a half transparent and half reflective optical mirrors, placed in front of the user's eyes, to capture the real world and display the virtual images at the same time. The real world is seen directly through the optical mirrors without modifications. This gives a natural and intact view of the real environment. The mirrors also reflect the virtual images to the user's eyes. A small projector projects the light beams representing the virtual images on the mirrors. Figure 8 shows the full process. The optical mirrors are called combiners because the final output is an optical combination of the real and virtual environments in a single view [8] [2].

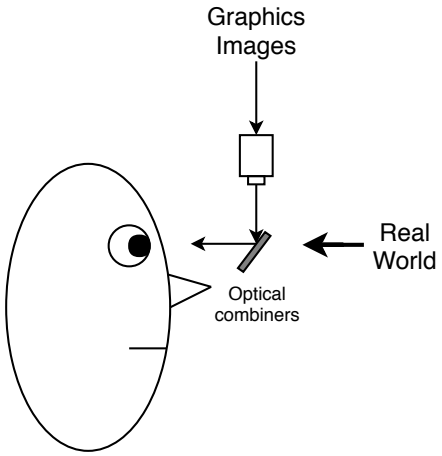

Figure 8. Optical see-through conceptual diagram.

The optical combiner is the key technology that enables mixing the real and virtual images for OST-HMDs. An ideal optical combiner is the one that is small, thin, bright, low cost and has a large field of view, low weight, high resolution, and low distortion. Companies also pay attention to the shape and look. Since nobody wants to look strange in front of other people, the optics and the whole display itself should be well designed. Unfortunately, such an ideal combiner does not exist; yet researchers are actively developing new solutions that improve the listed characteristics. The existing commercial combiners are good in certain characteristics only and fail to satisfy all the desired specifications. Below, we review the four main optical combiner technologies used in the industry nowadays.

Solid Beam Splitter is a basic type of optical combiner, used on e.g. SONY Glasstron, Google Glass, and Epson BT-300 OST displays[5] [13]. They are usually used with Liquid Crystal on Silicon (LCoS) microdisplays. The LCoS displays are very common in near-eye displays. In the solid beam splitter setup, they function as a transmissive and reflective component that generates clear images [5]. They receive light from an external source, such as a projection engine or a simple sequential RGB LED, and then transmit the light to the beam splitter. The beam splitter directs the light towards the display, then to the eyes. Figure 9 shows a simplified illustration of the technique. Usually, because of the weight and costs, the size of the beam splitter is small. This results in getting a very small field of view and the users may even see the blurry edges of the splitter [37]. One major drawback of the solid beam splitter technology is the brightness. It fails to generate dense light that can cope with natural light coming to the eyes from the environment. This makes it limited to indoor applications only.

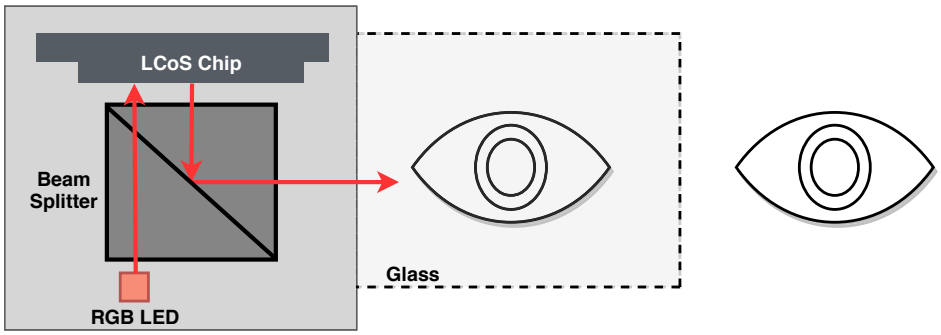

Figure 9. Illustration of light direction in a beam splitter setup. The light is first emitted from an RGB LED towards an LCoS chip, which forwards it to the beam splitter. The later directs the light to the display.

The second type of see-through optical combiners is **Spherical/Semi-Spherical Large Combiners**. A flat LCD or OLED is placed above one or two large combiners, to generate the display images. The large combiners reflect the emitted light to the user's eyes [37]. This technique is simple and supports high resolution and a big field of view. But the combiners are very large and usually take a big space. Meta 2 headset adopts this technique [38]. It uses two very large semi-spherical combiners with a single flat OLED panel on top of them. The overall resolution of the display is 2560 x 1440 which is split in half for the two eyes. Meta 2 supports up to 90° FOV. This is

considered as a very good FOV, and, in fact, incomparable with other small FOVs supported by the other AR headsets available on the market (for example, Google Glass has 13° FOV). Furthermore, compared to Solid Beam Splitter, this type of combiners is cheaper and easy to manufacture, due to their simple architecture and the wide availability and the low cost of LCD/OLED panels.

The third type is **Tilted Flat Combiner**. It is used by Osterhout Design Group (ODG) model R-7, as shown in Figure 10. The tilted plates receive light from the microdisplay and optics located just above the eye brow, and direct the images directly towards the user's eyes [37]. This simple architecture is one of the cheapest techniques used is optical see-through displays. It can support good image resolution and big enough FOV (about 37° on the R-7) still being light enough. However, any attempt to further increase the FOV requires more volume and space for the combiner. Because it is located near the eyes, increasing the volume brings the tilted plat closer to the eyes, which is considered highly undesirable.

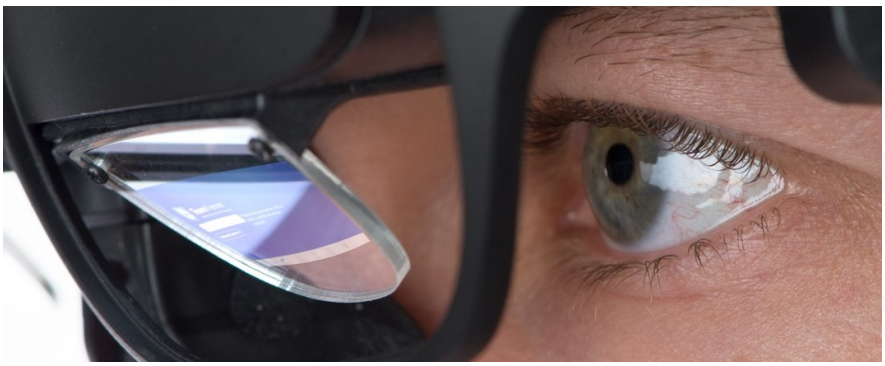

Figure 10. ODG R-7 model uses tilted flat plates to optically combine the real and virtual images. Credits: TeamViewer

The last type is **Flat Waveguides**. This is regarded as the most advanced technology for optical combiners up-to-date, and many believe that is the future of see-through optical combiners. This technology is based on a phenomenon known as Total Internal Reflection (TIR). Simply, when light enters the flat optical element, such as a glass or clear plastic, it starts reflecting internally [37]. Diffractive Optical Element (DOE) is used to transmit the light to the glass and also transmit it out of the glass. In the case of see-through combiners, light is emitted from a light source towards lenses, then it enters the glass via DOE. The light starts reflecting inside the glass until it reaches the part facing the eye. At this part, another DOE directs the light out of the glass in the direction of the eye. Figure 11 illustrates the principle. DOE is the key element of the flat waveguides technology. Without it, the light splits on all directions before entering the glass. Similarly, it is very hard to transmit sufficient amount of light to the eyes at the right place without the help of the DOE. Another kind of diffractive grating similar to DOE is Holographic Optical Element (HOE). HOE has been used as well in waveguide combiners [4].

Microsoft's Hololens combiner adopts waveguide technology [5]. It uses two HD 16:9 light engines to generate the virtual images, and utilizes DOE to blend the light inside a flat optical display and to push it back outside of it. The engines work like

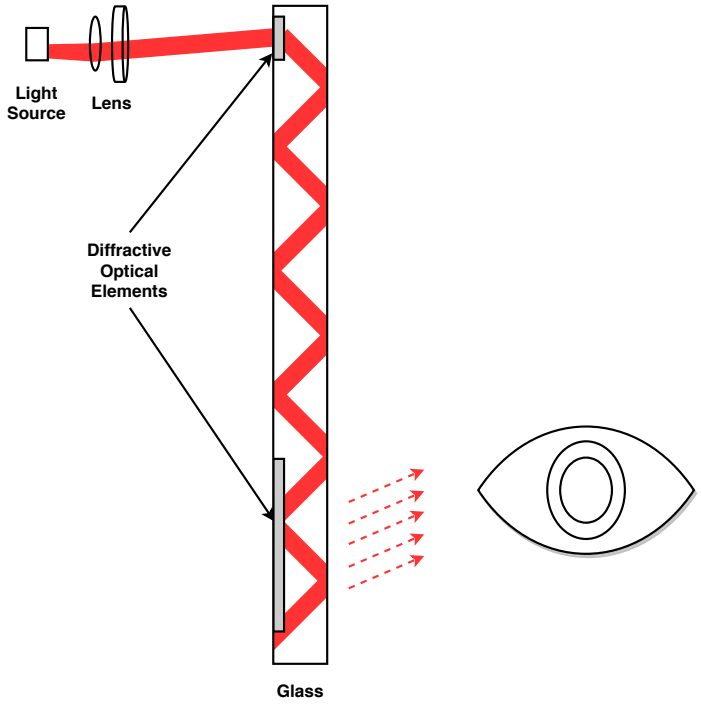

Figure 11. Advanced waveguides are built based on the TIR principle. Once light enters a flat optical surface it reflects internally. Diffractive optical elements are used to enter and exit the light from the optic.

near-eye projectors and emit the light towards the optical surface. Diffractive optical elements catch it and enter it to the surface. Once the light gets inside, it starts bouncing (TIR) until it reaches a triangular fold zone as shown in Figure 12. The fold zone causes the light to turn 90° down to a rectangular zone, where other DOEs are waiting to transform the light towards the eyes [37]. Hololens combiner is considered as a miracle of optical see-through technology. However, such a complex architecture have to be manufactured carefully which makes the mass production very complicated. This results in high production costs [39].

3.2. Optical Versus Video See-Through Displays

Video see-through head-mounted displays use a different strategy in combining the real and virtual worlds than optical see-through displays. In VST displays, a video camera placed in front of the device captures the real world. The obtained digital images, representing the real world, are combined together with the images representing the virtual world. Practically, the real images are set as a background and the digital graphics are aligned on the top of them. This combination is made inside a graphics processor. The output images are then displayed to the user using a simple monitor such as an LCD or OLED mounted in front of the eyes. Figure 13 illustrates the concept of video see-through displays.

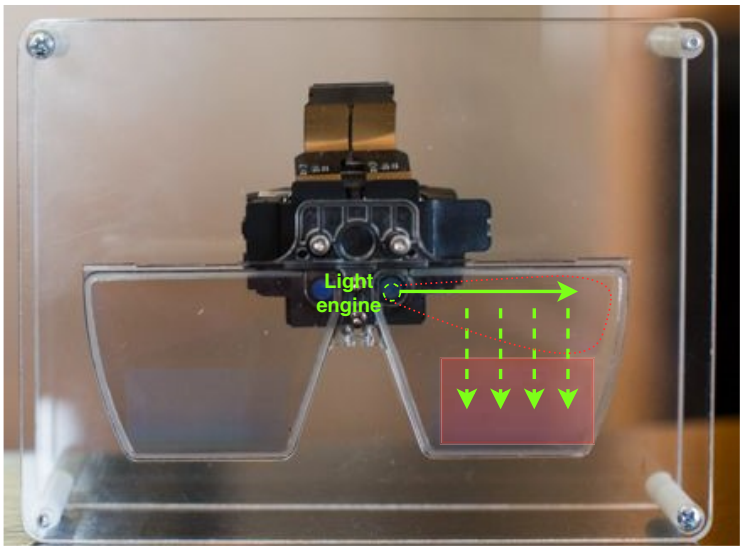

Figure 12. Hololens uses advanced waveguides for combining the real and virtual images. Image credits: https://www.theverge.com

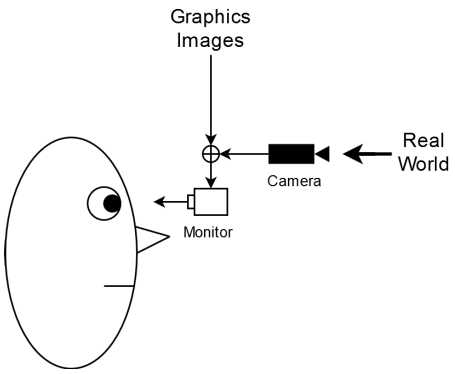

Figure 13. Video see-through conceptual diagram.

Video see-through HMDs are very good at merging the real and virtual scenes. Since the real world is captured and then treated as digital images, computer vision and image processing algorithms can be easily applied to process and modify the real scene. This pixel-based processing of the real scene gives a big advantage for VST displays to handle the occlusion problems [40]. Thus, they suit quite well for rendering X-ray visualization. In OST-HMDs, the real world is seen as it is. No modifications can be applied to change the scene. This makes the rendering of occlusion and X-ray challenging. In fact, it is very difficult for OST displays to occlude the real object using virtual objects only, due to the transparent nature of the optical combiners. The natural light goes through the transparent combiner directly to the eyes. As the light of the display is just combined with the natural light before serving it to the eyes, the natural light cannot be stopped or occluded simply by the display light. This makes occluding real objects with virtual objects very hard, if not impossible.

Another advantage of VST HMDs is that images of the real scene can be delayed by few frames until the virtual graphics are fully generated. As a result, the virtual objects can be easily rendered to appear in their appropriate places at the right time. The technique of artificial delay helps a lot in reducing the end-to-end latency as well [8]. However, no delay can be introduced to the real scene in OST displays. The rendering should be faster to present the virtual objects in their right places with respect to the environment. In X-ray applications, delaying the scene helps to create a better and convenient visualization. Since this is not possible with OST displays, applications implemented on these devices should be optimized as much as possible, so that virtual objects are rendered very close to their right places. The post-rendering and predictive tracking are key solutions for optimized systems [41] [42] [21].

The good advantage of optical see-through HMDs is that they leave the view of the real world as it is. Without any delay or modification, the real scene always appears natural. This property is necessary for many applications. On the other hand, the fact that you cannot change or delay the real scene requires that OST systems should have low latency. Most of the related work of AR X-ray vision has been done on VST displays. It would be very nice to see an X-ray system on an OST display that overcomes these challenges. This kind of a system is where we aim at in this thesis.

3.3. Latency Compensation for Optical See-Through Displays

In head-mounted displays, the time delay from tracking the user's head motion to presenting the rendered image to the user is known as the latency or end-to-end lag. It is a critical parameter for all AR headsets. For an acceptable AR experience, the latency should be kept below 60ms [20]. In OST-HMDs, when latency is big, registration errors, motion sickness, and disorientation are observed. This confuses the user [4]. In VST-HMDs, in which the captured real images can be delayed by few milliseconds until synthetic images are rendered in their correct places, as discussed previously, the lag can be reduced easily. To compensate for the latency, different approaches have been used. Extended Kalman filter (EKF) and other prediction filters have been extensively applied. Zhao et al. [42] proposed a method that helps to estimate the total latency as well as measure the effects of the compensation algorithms.

Kijima et al. [21] used image shifting techniques to reduce the latency. They have introduced a term Reflex HMD, based on vestibulo-ocular reflex, which can be applied to any HMD headset that rapidly updates the display image using raw head measurements without going through the whole rendering pipeline. In their prototype, they have used raw gyroscope data to change the viewpoint within the rendered image which is larger than the screen resolution. The viewpoint is shifted according to the measured head movement. This shift is made using hardware to manipulate the driving signal of the LCD displays, as shown in Figure 14. Their experiments showed that using cylindrical rendering helps in improving the overall performance through deformation cancellation and distortion decreasing. Oculus Rift uses a similar approach using a high-speed IMU and pixel resampling hardware [4]. Similarly, Microsoft Hololens mixed reality headset uses its own hardware compensation, called image stabilization pipeline. It is build based on the algorithm of Reflex HMD. Using the stabilization pipeline, Hololens corrects for the head movements, occurred after the scene rendering, and reduce the

overall latency. When programming apps for the device, developers have access to the hardware. They can define a single plane in the 3D scene that will receive the maximum hardware stabilization. This plane is called **image stabilization plane**. In this thesis, we have used efficiently the Hololens' image stabilization hardware.

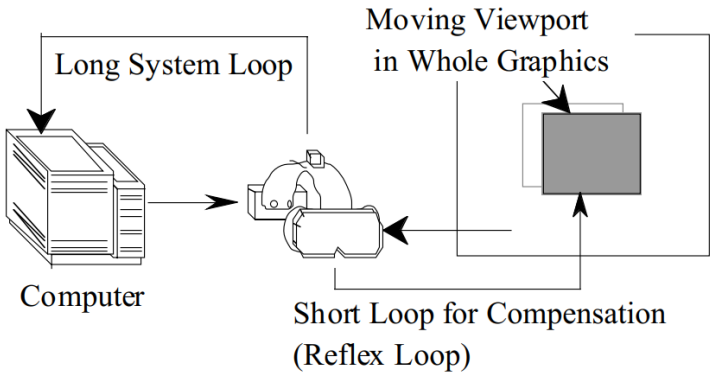

Figure 14. Reflex HMD shifts the view using the raw measurements of head movements [21].

3.4. Commercial Headsets

In this section, we briefly look at the main commercial optical see-through displays available on the market. We present first Microsoft Hololens, the headset used in our work. Then, we describe other similar devices including Meta 2, Google Glass, and Epson BT-300.

3.4.1. Microsoft Hololens

Hololens is a stereoscopic optical see-through head-mounted display, developed by Microsoft and introduced in 2015. The mixed reality headset (Figure 15) is a small computer running Windows 10 operating system. It is able to sense the environment, track the user's pose, and augment the real environment with virtual objects. It is completely wireless and mobile, allowing users to freely move around. All the computational tasks are done in the headset itself, so no external components are needed. For the display technology, the headset uses waveguide technology to display high-definition images. The waveguide elements face the eyes and deliver a pleasant augmentation of the real world. The optics allow to see the real world as it is and, at the same time, they display the virtual objects/creatures in their accurate places with respect to the real environment. Interestingly, the objects stay stable when moving or just rotating the head. This stabilization and registration are mainly achieved due to the high quality tracking used by the device. The headset uses four RGB cameras dedicated for optical tracking as well as a depth camera and an IMU. The users can interact with the content displayed by the device using gestures, gaze, and voice recognition. Hololens provides also interesting features like 3D sound and spatial mapping.

The spatial mapping provides a detailed 3D mesh map about the surfaces of the surrounding environment [43]. Table 1 gives more details about the specifications of the device.

Developing Windows 10 apps for Hololens is mainly done using Unity 3D or DirectX platforms. Unity 3D is the most common platform among developers. The Windows APIs support apps written in C++ or C#. Nowadays, Microsoft is targeting companies by presenting the Hololens as a great data visualization and productivity tool. Moreover, the company announced they are developing a new generation, Hololens 2, that is expected to outperform its predecessor, and it will include a custom AI chip dedicated for running neural networks and AI algorithms.

Figure 15. Hololens headset.

Table 1. Microsoft Hololens characteristics.

Feature	**Specification**
Ocularity	Binocular
Image Source	LCoS Microdisplays
Resolution	720p / 2HD 16:9 light engines
Color	Color Sequential RGB
Field of View	30°
Head Tracking	1 IMU
	4 Environment Understanding Cameras
	1 Depth Camera
Camera(s)	4 Environment Understanding Cameras
	1 Depth Camera
	2MP Photo / HD Video Camera
Main Processor(s)	Intel Atom x5-Z8100 1.04GHz
	Intel Airmont (14nm)
	Custom Holographic Processing Unit (HPU)
RAM	2 GB
Development Environment	Windows 10, 32bit

3.4.2. Meta 2

Meta 2, shown in Figure 16, is an AR headset developed by the company Meta. It was launched in 2016. With 90° FOV and 2560x1440 pixel resolution, Meta 2 delivers the world's most immersive AR experience [44]. It uses a very large combiner with a flat high definition OLED panel. The combiner is transparent, and thus allows natural light to come to the eyes. The headset is connected to a desktop computer via a cable. Actually, most of the computing tasks are done outside the headset. This gives a significant computing power to the device, but limits its mobility. More features of Meta 2 are provided in Table 2. App development for Meta 2 is done using Unity 3D platform. The Meta company provides a Unity SDK to build and share the apps. The SDK includes SLAM, hands interaction tracking, example code, and documentation [45]. Though Meta 2 is known to have a great display and immersive AR experience, it has problems with tracking. The virtual objects are not that stable as in Hololens. This means the tracking quality of Meta 2 is not as good as the one of the Hololens, and it can be better.

Figure 16. Meta 2 headset. Credits: Meta.

Table 2. Meta 2 characteristics.

Feature	**Specification**
Ocularity	Binocular
Image Source	OLED
Resolution	2560 x 1440
Field of View	90°
Camera(s)	720p
Development Environment	Windows 10

3.4.3. Google Glass

Google Glass, shown in Figure 17, is a monocular optical see-through head-mounted display, developed by Google and manufactured by Foxconn. It was first released in 2013, then pulled out from the market in early 2015. The Glass uses Android as an operating system and split combiner technology for displaying digital images. The users can interact with it using gestures, touchpad, and voice recognition. The touchpad is on the side of the device and allows users to slide through apps. Examples of voice commands are "take a picture", "record a video", "make a call to (name)", etc [12]. Overall, Google Glass is lightweight and has a neat design. It is very suitable for applications that contain information display and data visualization. Google Glass created new research paths and enabled new innovative applications [5]. The applications targeted hospitals, industry, and warehouses. Google announced that a new glass is under design and development. Details about the main features of the Glass are presented in Table 3.

Figure 17. Google Glass display. Credit: Tim Reckmann, Wikimedia.

3.4.4. EPSON BT-300

The BT-300 from Epson, shown in Figure 18, is the third generation of Epson's Moverio smart glasses, introduced in 2016. It is a binocular optical see-through AR display based on split combiner technology. It has a slim, neat, and lightweight design. The first generation, BT-100, was launched as early as 2011. After that, the Moverio family has seen main improvements in terms of design, optical thickness, brightness, FOV, and resolution. Epson developed its own display technology that provides higher resolution and higher brightness, called Si-OLED [13]. It uses Silicon-based OLED to generate digital images, then projects them to the split combiner. The BT-300 display

Table 3. Google Glass characteristics.

Feature	Specification
Ocularity	Monocular
Image Source	LCoS Microdisplay
Resolution	360p
Color	Full Color
Field of View	15°
Head Tracking	9 DoF IMU
Camera(s)	5MP Photo, 720p Video
Main Processor(s)	OMAP 4430 SoC, dual-core
RAM	2 GB
Development Environment	Android 4.4

adopts this technology. It provides dual 1280x720 pixels per eye and good visibility in bright sunlight, which makes it suitable for outdoor applications. The smart glasses has about a 23° FOV and considered as one of the lightweight displays on the market (headset weight is 69g) [5]. BT-300 uses Android 5.1 operating system. With a good processor and 2GB of RAM, the glasses can run computationally expensive and demanding apps. In addition, the device supports WLAN, so it can be connected to the Internet or wireless local networks. A small hand controller is attached to the glasses. The controller has three programmable buttons, a circular trackpad, and a rechargeable battery. Most of the computing tasks are offloaded to it. Table 4 summarizes the main features of the glasses.

Figure 18. EPSON BT-300. Credit: Epson.

Table 4. EPSON BT-300 characteristics.

Feature	**Specification**
Ocularity	Binocular
Image Source	Proprietary Silicon OLED (Si-OLED)
Resolution	1280 x 720
Color	24-bit color %
Field of View	23°
Head Tracking	1 IMU
Camera(s)	5MP camera
Main Processor(s)	Intel Atom 5 1.44GHz Quad Core
RAM	2 GB
Battery Life	Approximately 6 hours
Development Environment	Android 5.1

4. X-RAY VISUALIZATION

X-ray vision refers to the ability to see through walls and surfaces. With the help of computer-generated images, users can see the hidden parts behind occluding surfaces. Usually, it is described as the superman's ability to see through walls and buildings. X-ray visualization is considered as an important application of augmented reality. During the last two decades, many works have been presented to slove the issues related to it. From time to time, researchers propose from time to time new methods and solutions that push the quality of the visualization further. In this section, we review the state-of-the-art of the augmented reality X-ray. In addition, we discuss as well some of the used methods to scan 3D environments and spaces, in order to be used later in X-ray applications.

4.1. Augmented Reality X-ray Vision

The introduction of OpenGL stencil buffer allowed to look beyond occluding objects in pure 3D scenes. Coffin and Höllerer [46] used stencil buffer techniques to perform cutouts in surfaces that occlude other objects. Their work facilitated AR X-ray vision. In the presented system, the user starts defining holes on the occluding surface as well as a point of interest that points to a hidden object to be seen. Then, based on the holes and the position of points of interest, a cutaway is performed automatically on the surface to allow seeing through it. All the information of the occluding surface falling within the cutaway area is removed from the view. The algorithm uses the surface's normal vector, to orient and reposition the cutaway geometry, so that it is rendered with respect to the user's view. That means, even if the user turns around or moves from his initial place, he still can see the correct X-ray cutout. Figure 19 shows the results of this technique.

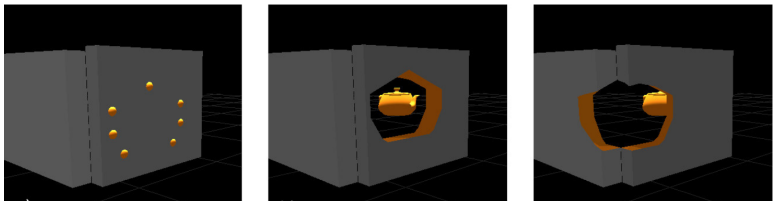

Figure 19. The user defines initial holes, then a cutaway is performed based on the hole positions [46].

Simply rendering the virtual images that represent the hidden objects on top of the occluder may confuse the user. Rather than seeing the objects behind the occluder, the user gets a feeling that the hidden objects are floating in front of the occluder. Thus, X-ray systems should be designed to give correct occlusions. In [47], the authors improved the special awareness of AR X-ray systems using an edge overlay from the occluding surface. The edge overlay is computed from images of the occluding surface. Images provided by a camera behind the occluder are combined with the edge overlay. The resulting X-ray visualization provides a good foreground context. This

technique produces additional depth cues and helps to get the right relationship between the occluder surface and the occluding objects.

Sandor et al. [23] took one step further by proposing an X-ray system based on visual saliency. The system builds a map of salient features to detect important visual landmarks. Salient features include any color, luminosity, orientation or motion that attract visual attention. Edges from the occluder surface are considered as one salient feature. The proposed system displays important visual information, through rendering these visual landmarks of both the occluder and the occluded objects. Such information helps to perceive depth and recognize the correct relationship between the occluder and the occluded. The authors highlighted that occlusion is the most important depth cue, and the other depth cues depend on it. They have presented a diagram, shown in Figure 20, that demonstrates the relationship between the different depth cues based on the work on [48]. They also claim that rendering the important regions based on the saliency maps preserves correct occlusions.

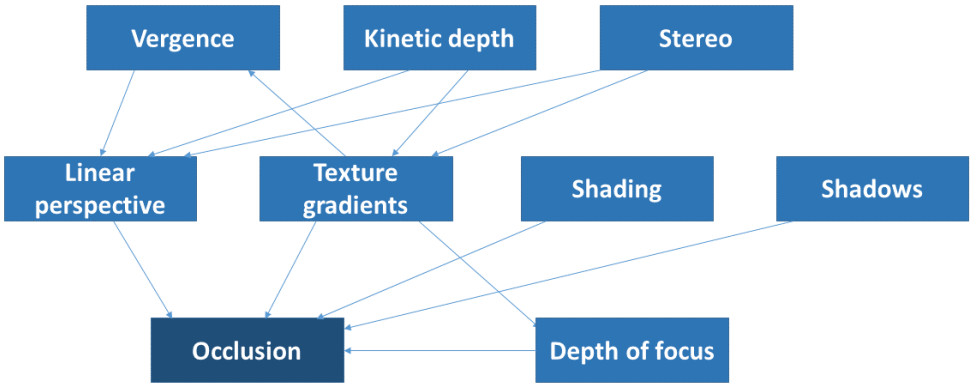

Figure 20. The relationship between different depth cues.

Kameda et al. [49] presented an outdoor mixed reality see-though system, based on a number of surveillance cameras embedded in the environment. The system is implemented on a handy personal computer (HPC) that has a back camera attached to it. The computer displays the live status of the invisible areas that the user cannot see due to big structures, such as walls and buildings. The surveillance cameras are located in different positions. They capture the live images of the hidden areas, then stream them to the computer, which uses them as a source of the hidden areas. The system ameliorates the results using pre-created CG models. The models represent the static area and have been created using CAD tools. The system outputs a precise visualization where the images of the hidden areas are imposed on top of the images of the computer's camera. Figure 21 shows an overview of the algorithm. All the surveillance cameras are calibrated against the world offline, to get a better estimation of their locations and

orientations. Since it is difficult to place markers in outdoor environments, some parts of the buildings are used as markers for the calibration, together with the CAD models.

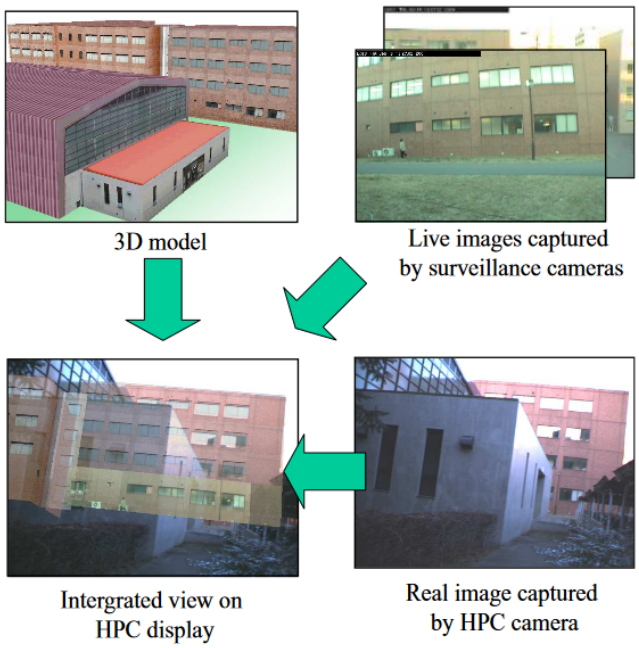

Figure 21. Information from three different sources is used to render the X-ray vision [49].

One application of X-ray is to visualize underground networks, such as water and gas pipes. Jiazhou Chen et al [50] introduced an X-ray system to visualize underground structures. Their approach uses video processing tools, to add virtual objects, representing the hidden structures, to the video frames. The system does not require pre-scanned 3D models of the environment and assumes that the capturing camera is located in a fixed position. The system extracts features from the frames to create visual cues, which help users to understand the occlusion relationships. The extracted features, like edges and second order derivatives from images, are displayed completely, or used with blending masks. The fact that no prior 3D information about the area is needed, facilitates the implementation of the X-ray visualization because there are no misalignment problems between the 3D model and the reality to handle. The size of the filters of the edge detection and transparency smoothing affects greatly the total frame rate. The authors reported that the frame rate of the system ranges from 12 to 18 fps. We believe that higher frame rate can be achieved with the current hardware, especially if using parallel computing methods.

An augmented reality X-ray that uses Google Street View to get images of the hidden scene has been presented in [51]. The prototype was designed for outdoor cases. It displays the X-ray visualization on the facade of buildings. The system was implemented as an Android mobile application. The proposed application adopted techniques used by previous X-ray systems such as using edge overlay to provide depth cues. In addition, the authors introduced a new silhouette computation method to provide more visual information about the occluding surface. This improves the projection of the

images of the hidden scene into the real surface. A brightness adjustment step is performed to deal with the outdoor brightness issues. In order to cope with the limitation of computation capabilities of mobile phones, the images are downscaled before being processed by the edge detection and silhouette computation algorithms. After the processing, the images are rescaled to their original scale again.

4.2. X-ray Vision on Optical See-Through Displays

Most of the AR X-ray systems have been made for VST-HMDs and handheld devices. In this section, we review the systems that have been implemented on OST-HMDs. Julier et al. [52] presented the implementation of an outdoor AR system that provides situational awareness information to warfighters. The Battlefield Augmented Reality System (BARS) is designed to be used in urban environments, where there are a lot of obstacles that block the view, such as buildings and facades. The whole system is composed of a mobile computer, a tracked optical see-through HMD, as well as a wireless communication system. The system shows only the most relevant information on the current situation. Important objects in the urban environment are symbolically presented to the user. The user can see for example the wireframes of buildings, important targets, the names of streets, sniper locations, and so on.

Livingston et al. [53] studied different display attributes for X-ray applications involving far-field occluded objects. The authors used an OST setup with purely virtual occluded objects. The aim of the study was to discover the graphical cues that help the user to maintain the correct depth relationships of virtual objects. They designed three different X-ray visualization techniques for outdoor environments and performed a user study to reveal which visualization is the best in expressing occlusion relationships. Each visualization assumes four layers of occlusion, three buildings, and a target. Each layer is located at a different depth.

A recent X-ray vision system was implemented by Rompapas [54] on OST devices. The system requires two users both wearing Google Glasses. One of the Glasses streams the poses and live video images to the other Glass which uses the received data to perform the X-ray visualization. The system works only if one of the users sees the point of interest, while the other one cannot see it because of a blocking surface, like a wall, for example. Another great feature of the proposed system is that it allows the users to move around freely.

4.3. 3D Scanning for X-ray Vision

For the X-ray visualization, information about the hidden objects is needed. One way to represent it is to use a pre-built 3D model of the hidden area. The idea is to scan the area offline, then export the data into a 3D model format. Usually, mesh algorithms are used to optimize and modify the model, so it will be made suitable for the application. Later, on the runtime, the X-ray system uses the 3D model to represent the unseen parts of the scene. In this section, we cover few techniques used to 3D scan environments.

Structure from Motion (SfM) is one approach to determine the 3D structure of spaces and objects using cheap cameras. The technique uses a set of 2D images to estimate

the 3D structure of the scene. Usually, the object or the area to be scanned is in a fixed position, while the camera taking images moves around it continuously. SfM is a well-known topic in computer vision, and it has been widely studied and developed by researchers. Current Structure from Motion implementations are often based on local keypoints and descriptors such as Scale-Invariant Feature Transform (SIFT). [55]. First of all, features are extracted from each image using the SIFT algorithm, for example, and a descriptor is created for every detected feature. Then, the descriptors are compared with each other to reveal the overlappings. The overlappings help to link the images with each other. This is called matching. When all the images are matched, the overall structure of the 3D scene is obtained. One free and easy to use software to go through these steps is VisualSFM. That is a popular desktop application among scholars published by Changchang Wu [56]. It uses SfM algorithms to generate 3D point clouds of objects or scenes taking 2D jpeg images as input. The software supports parallel computing and integrates several reconstruction algorithms developed previously by the author, including Multicore Bundle Adjustment [57], SIFT on GPU, and Towards Linear-time Incremental Structure from Motion [58]. The basic pipeline used by the application is as follows: add images, match images, do a sparse reconstruction, and then a dense reconstruction. The final output is a point cloud in .ply format, representing the 3D geometry of the environment or object captured by the 2D images. The .ply files created by VisualSFM can be converted later into textured 3D meshes using Meshlab [59], for example.

Another easy-to-use method to obtain 3D information from spaces is Matterport Scenes. Matterport Scenes is an Android app developed by Matterport, Inc. It runs exclusively on Google Tango devices: ASUS ZenFone AR and Lenovo Phab2 Pro [60]. The app uses available camera sensors and Google Tango APIs to generate 3D point clouds of the surrounding environment. All the processing is done on the smartphone internally, without using GPS or external support such as cloud-based services. The user can export the 3D data as a .ply file, and use it later for other applications. Moreover, there exist other iOS-based scanning apps for Structure Sensor [61]. That is a 3D sensor designed for iPads only. It can capture depth data as well as RGB images.

Matterport has a 3D camera as well, called Matterport PRO2 [62]. The camera is rather expensive, and thus, used mainly by professional purposes. It gives very accurate and dense results and can scan big areas, including whole buildings. The camera captures color and depth, then sends them to be processed in the cloud. The user has to pay both for the camera and the cloud processing. Recently, a new dataset of point cloud scans from Matterport PRO2 3D Camera has been published [63]. The scans have been made in five big areas including offices, educational spaces, conference rooms, meeting rooms, and more. Every point of the point clouds is labeled according to 12 categories (floor, wall, window, chair, table, etc.). The dataset allows students and researchers to develop algorithms that deal with 3D point cloud scans.

The last covered scanning method is Bundlefusion. It is a real-time scanning framework published by Dai et al. [64]. It uses one RGB and one depth camera to extract 3D mesh of the space. It supports several devices that have RGB-D sensors, such as Microsoft Kinect, Intel RealSense, and Structure Sensor. The framework processes the scene and generates the 3D mesh in real-time. It uses advanced parallel computing methods, such as Cuda and requires a good GPU. No offline processing is needed. The algorithm takes RGB-D frames as input and then detects correspondences between the

input frames. The algorithm uses a GPU-based SIFT feature extraction, matching, and pruning. Then, it performs a combination of local and global alignment steps. It uses sparse and dense correspondences for the pose estimation. The result obtained from every frame is added to the previous results. By doing so, the reconstructed 3D model updates continuously. The framework supports relocation and tracking recovery. That is, when tracking errors happen, it can recover and relocate the camera with respect to the scene. After the recovery, the algorithm continues scanning.

5. IMPLEMENTATION

This chapter describes the details of the implementation of our augmented reality X-ray system. We have implemented an X-ray cutout algorithm, as well as four different visualizations that use it. First, we start by reviewing the software and platforms used for the development of the system. We cover Unity 3D, Vuforia SDK, and OpenCV. Next, we present the technique used to create a 3D model of the real scene. The model is needed for both the cutout algorithm and the visualizations. Finally, we explain our X-ray cutout algorithm, followed by the four different visualizations.

5.1. Platforms

In this section, we give details about the platforms used in the implementation of our work. First, we review Unity 3D, the software that hosted all the development. Unity 3D engine supports making applications for augmented reality headsets, including Microsoft Hololens. Then, we describe the augmented reality SDK, Vuforia. It is one of the most popular AR platforms. It implements different computer vision algorithms that help to develop AR applications. Lastly, we cover OpenCV, the powerful and very optimized open source computer vision library. It provides multiple computer vision algorithms used for face/object detection, camera calibration, stereo vision, feature detection, classification, and more.

5.1.1. Unity 3D

Unity is a cross-platform engine for making games and 3D interactive applications. It is developed by Unity Technologies. The first version was launched in 2005 and supported MacOS only. Through the successive released versions, it was gradually extended to support several platforms. Now, Unity can deploy games and applications to more than 27 platforms, including Windows, Mac, mobile devices, and video game consoles. It has a free and a commercial versions that include more features. Unity makes it possible to start creating 3D interactive applications and video games without big investments. It offers interesting features like real-time lighting, 3D coordinate spaces, 3D physics engine, audio, dynamic shadows, and more. For scripting, C# or JavaScript are used, but most of the Unity developers prefer C#. Unity is mainly used for creating games, but it can be used to create 3D simulations, animations, visualizations, etc, as well. Moreover, Unity provides native support for different commercial AR and VR headsets, including Microsoft Hololens. Developing applications for Hololens requires Unity and Visual Studio. First, the application is created with Unity, then built and deployed to the device using Visual Studio. The deployment can be done via a USB cable or wirelessly. Microsoft and Unity work closely to deliver unique features related to the Hololens and other immersive headsets. These features are used by developers when making applications using Unity. These include gaze, gestures, voice command, 3D audio, world coordinates, spatial sound, and spatial mapping. In our work, we have used Unity 2017.02 and Visual Studio 2017 to build and deploy our

X-ray system to the Hololens. More details about the development of our system are discussed in the next sections of this chapter.

5.1.2. Vuforia SDK

Vuforia is an augmented reality SDK created by Qualcomm Inc, now acquired by PTC. The SDK gives mobile application developers a rather easy way to create AR experiences without significant efforts to develop the required computer vision algorithms. It facilitates the work for developers by providing these algorithms. Users can detect targets on the real space and track their pose in real-time. The SDK can be used together with the main app development software, including Android Studio, Xcode, Visual Studio, and Unity. Thousands of apps powered by Vuforia have already been published in Google Play and App Store [65]. Furthermore, the SDK supports HMDs that can be programmed using Unity, such as Microsoft Hololens. Development of basic applications for Unity projects is free, but includes the Vuforia watermarks. The recent versions of the Unity integrated Vuforia within the editor. For more advanced features, like storing targets on the cloud, developers need a commercial license. Also, the license is needed for all native development (iOS, Android, UWP) rather than Unity, including even the basic features.

Vuforia platform uses images as targets to launch an AR experience. Instead of scanning barcodes or QR codes, simple images can be used to start the experience. When the app starts, the platform creates a global coordinate system. Whenever an image target is detected on a given image frame, the platform registers its pose with respect to the coordinate system and tracks it. The AR experience is launched along with the detection. The experience can start an animation, place a 3D object on top of the image, play a 2D video, etc. The image targets are stored in a database locally on the device. Users who have the license have the possibility to store them on the cloud. The SDK comes up with several predefined targets. Users can print and use them directly in their applications. However, if they want to use their own image target, they should scan, measure and add it to the database that Vuforia uses on the runtime. It is very important to provide accurate measurements of the image, otherwise tracking problems will occur.

On our AR X-ray vision system, we used Vuforia 6.5 with Unity to position and align the virtual 3D model of the scene with the real world. We printed out one of the image targets included in the SDK and attached it to a wall in a room, as shown in Figure 22. When the headset comes closer to the wall, and the camera captures the target, the 3D modeled wall comes into view, placed and aligned with respect to the real wall. The 3D model is linked with the image target in advanced in Unity. When Vuforia detects the target during runtime, the pose of the 3D model is initialized according to the target's pose. Vuforia uses the camera resource of the Hololens on its own and does not allow other parties or APIs to use it directly. If an API needs camera images, it should ask for them from Vuforia. The only way to access the images is via Vuforia API. This creates an unnecessary delay and limits some image processing capabilities. Since the main reason to use Vuforia in our system is to initialize the pose of the 3D model, after a few seconds after the target detection and pose initialization, we keep the 3D model in its place and stop Vuforia API. That releases the camera resource so that other

APIs and image processing libraries can use it, eliminates unnecessary computational power, and increases the frame rate of the application making the app to run faster.

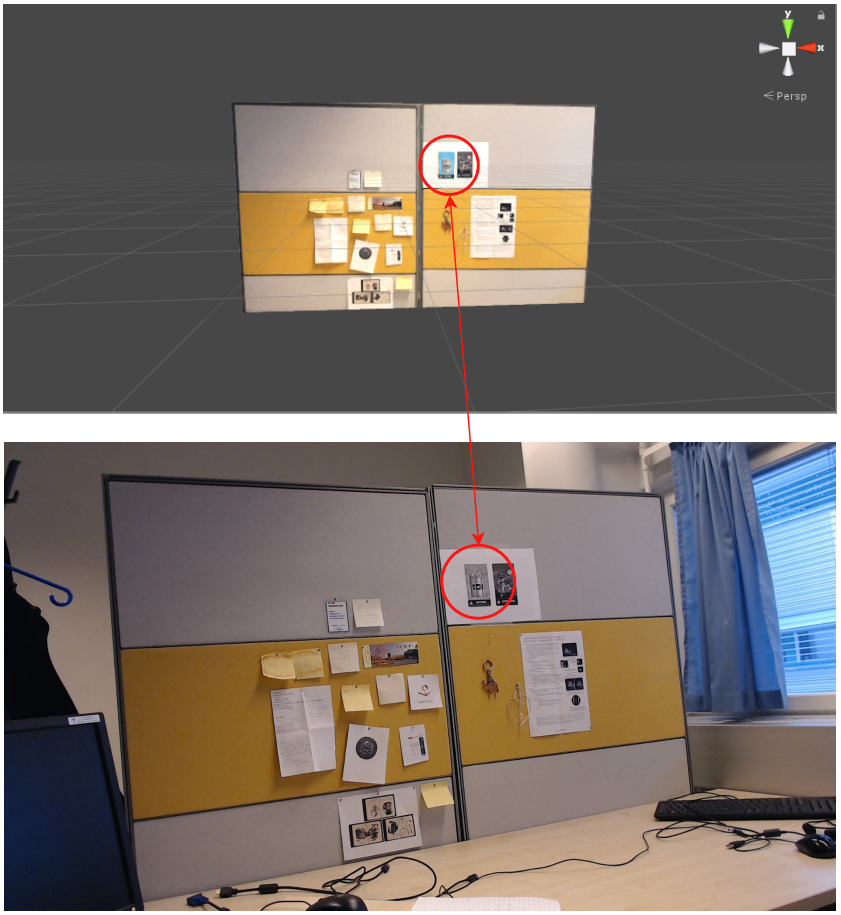

Figure 22. Vuforia image target is used to link the real and virtual worlds.

5.1.3. OpenCV

OpenCV (Open Source Computer Vision) is an open source library written in C/C++ for image and video analysis. It is designed for real-time applications and offers more than 500 optimized algorithms related to the field of computer vision. It was originally developed at Intel and introduced in 1999. The first version was launched in 2006. In 2009, the second version, called OpenCV 2, was released bringing important features and changes and the library is still continuously expanding. When writing this sentence, the latest version of OpenCV is 3.4.1 (February 2018). The library is released under a BSD license so that it can be used by researchers and developers free of charge. OpenCV has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS, and Android. Bindings and wrappers are available as well to support other programming languages such as C#. In order to use OpenCV in Unity projects, there is one asset in Unity Asset Store called OpenCVforUnity [66]. It is built based on OpenCV Java, allowing users to use the same API as the latest version of OpenCV

Java. The asset works on many hardware including Microsoft Hololens and can be used in applications related to augmented reality technology too. In this thesis, we used OpenCVforUnity to process live images taken by the Hololens in real-time to create a dynamic X-ray visualization. That is discussed in more detail in Section 5.4.4.

5.2. 3D Modeling of the Real Scene

The work presented in this thesis demonstrates the development of an AR X-ray vision algorithm. In addition, four different visualizations that use this algorithm were implemented. We evaluate them in the next chapter, and rank them regarding to the quality of the X-ray visualization and depth perception based on user experience. In this section, we present the 3D modeling of a real scene used to develop and test our X-ray vision algorithm, as well as the proposed visualizations. To simplify the tasks, we modeled a single wall in an office rather than the whole office. The wall serves as the occluding surface. We add later virtual objects behind the wall to test whether the cutout succeeded to visualize the X-ray vision. For a more convenient application, it is better to scan the whole office, using one of the approaches discussed earlier in the previous chapter. That would use the same algorithm developed in this work giving in addition a more realistic view. The straightforward technique to model the wall is to use the built-in Cube object of Unity and modify its size and scale to match with the real size of the wall. The model is textured manually using a color image of the real wall. We start by measuring the size of the wall using a handy measuring tape. Next, we create a Cube object in the Unity editor. We change its size and scale to match with the dimensions of the real wall. Lastly, we texture the wall manually by taking a picture of the wall, importing it to the Unity project and assigning it to the model. Several tests were then made to make sure the virtual model aligns with the real wall. The final model is presented in Figure 23.

5.3. X-ray Cutout

In this section, we discuss our X-ray cutout algorithm. In order to understand the implementation, it is necessary to understand how objects work in Unity. Unity uses a modular component system to construct the scene objects. All objects on a Unity project, including cameras, lights, 3D models, characters, and so on, are called game objects. Each game object contains several components attached to it. They decide how the object looks and behaves. This approach is not similar to the traditional inheritance architecture where a strict hierarchy of classes is used. In this component system, objects are in a flat hierarchy, and every object has multiple components that have different functionalities. By default, a Transform component is attached to all game objects automatically. That determines where the game object is located, and how it is rotated and scaled. All the components can be removed except Transform because all the objects must have a position, rotation, and scale in the 3D world.

Moreover, there are two key components that can be attached to game objects; scripts and renderers. Scripts decide how the game object behaves, while renderers decide how the object should appear. Unity refers to the code files as scripts. These can be

Figure 23. (Top) The real wall. (Down) The 3D modeled wall.

used to control the behavior of the objects, read input data, toggle graphical effect, send data to the Internet, implement image processing algorithms, and more. Developers can use C# or JavaScript to write the scripts. In order to run a code file, it should be a component of a game object. Scripts that are not attached to any game object are not executed. Unity does not allow linking to external code libraries, and therefore ,all the scripts should be inside the Unity project.

Commonly, 3D models in Unity projects have a MeshRenderer and MeshFilter components. The MeshFilter tells what is the shape of the object, while the MeshRenderer, called also renderer, tells how the object should appear. A game object can have only one renderer, but the renderer can have multiple materials. A material is a simple wrapper for a shader. The diagram in Figure 24 shows the relationship between these components. The material can store properties and textures and use them to control the shader. Multiple materials can be created based on a single shader. The shaders are common programs in computer graphics used to determine how to draw the object, taking into account the light sources around it, material properties, the location of the viewer, surface orientation, and more. These are used by the GPU when calculating the color of each pixel representing the model within the final rendered image. Nowadays, shaders can even be used to create graphical effects or do video post-processing.

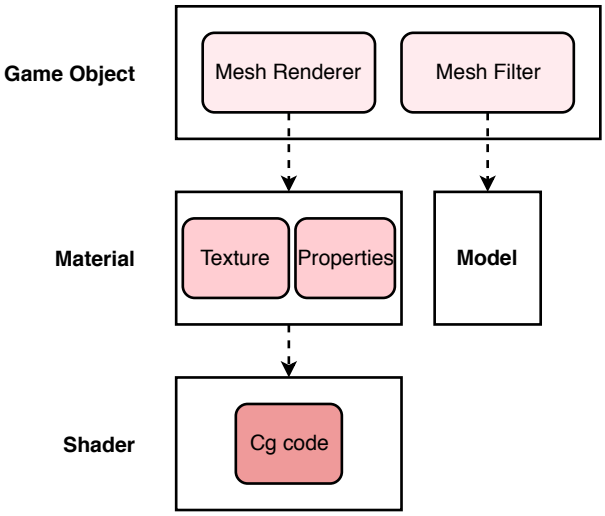

Figure 24. The relationship between game object, material, and shader.

Our X-ray algorithm is implemented on a shader file. A material is created based on the shader to control it and then attached to the 3D model receiving the X-ray cutout. The main function enabling the X-ray cutout within the shader program is the intrinsic HLSL function clip(). HLSL (High-Level Shader Language) is a shading language developed by Microsoft and can be used in Unity. The function is used to discard a certain pixel at a certain condition. It is usually used to simulate clipping planes and to test alpha behavior. It works in a similar manner to the stencil buffer techniques used by Coffin and Höllerer [46]. In our implementation, we have used the function to take away a 3D volume having the shape of a sphere from the 3D model. The volume is selected by specifying a position in 3D space and a radius. We discard the pixels within the selected volume and do not draw them. That illustrates opening a window in the model so that users can see what is behind it. Since shaders run in the GPUs, the algorithm benefits from the high-speed computing power of the GPUs. If the algorithm is implemented on C# scripts, it will take longer time to execute.

The discussed technique is efficient when applying it to a planar surface. To make our algorithm more robust, it is more convenient to make the cutout of a shape of a cylinder. This is because we are expecting to deal with a 3D model that may have a complex shape and/or take a lot of space. However, performing a simple sphere cutout in the middle of the model is not enough. The parts of the model that lays between the user and the cutout position can occlude the cutout. The solution is to use a cylindrical cutout. It ensures that a window is opened in the whole 3D model, not just a simple sphere cutout in a given position on the model. We mimic a cylindrical cutout by performing several spherical cutouts continuously. The cutouts have a shape of a sphere close to each other and aligned in a single line. We call the position of the first sphere the starting position and the last sphere the stopping position. These two positions determine the starting and ending positions of the cylindrical cutout. The rest of the positions are automatically generated. Figure 25 illustrates the implementation.

For a smooth rendering, the algorithm needs to update the starting and stopping positions continuously. We use a C# script to pass the two positions to the shader for

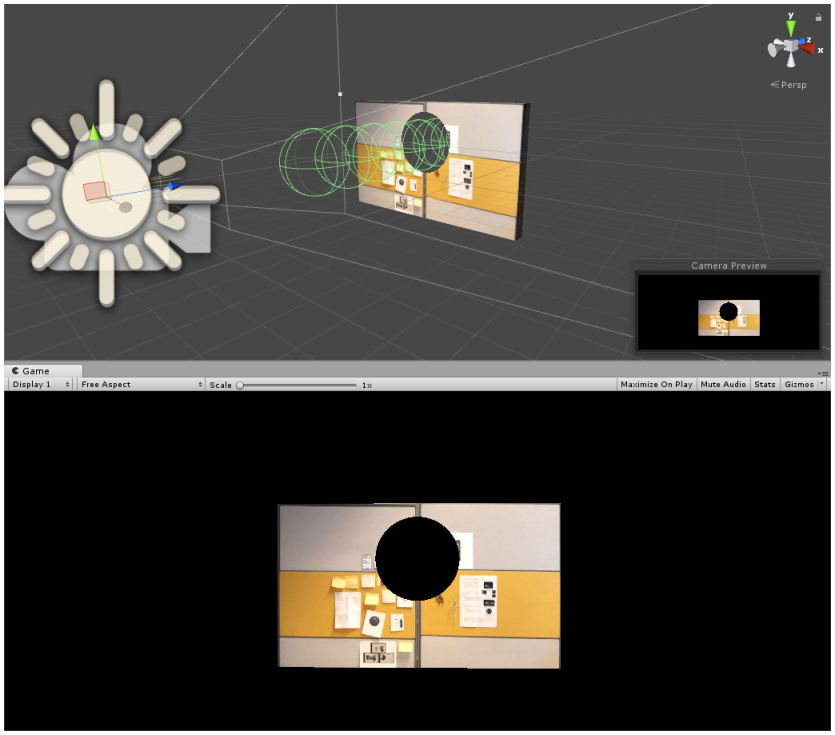

Figure 25. A cylindrical cutout is made with several spherical cutouts.

each frame. We call it X-ray Manager. The script calculates the two positions first, then sends the values to the shader. The cylindrical cutout is performed based on the received values. We assign the starting point to the current user's position, while the stopping point to a position that is in the same direction of the user's gaze in a distance of 3 meters. When the user moves his head or changes his gaze direction, the new positions are updated quickly. With the same script, the radius of the spheres can be changed, as well as the distance between every two successive spheres. We have used a custom lightning model as well to make the shader a lightweight and suitable to be used in devices that have limited hardware, such as the Hololens. The used lightning model is called Lighting Mobile Blinn Phong.

5.4. Visualizations

In this section, we show four different X-ray visualizations, implemented for the Hololens device. Each visualization uses our X-ray algorithm, and they are designed to help the user to estimate the depth of different objects correctly. The main difference between the visualizations is how to visualize information about the occluding surface. The visualization techniques are listed below and described in more detail in the following sections:

1. *Simple X-ray cutout*. This is the simplest visualization. No information about the occluding surface is presented. The user estimates the depth of the virtual augmentation using parallax only.

2. *X-ray with static edge overlay*. An edge overlay of the occluding surface is displayed on top of the occluding wall. The edges are extracted from the image texture of the 3D model.

3. *X-ray with dynamic edge overlay displayed on a 2D plane*. An edge overlay based on live images taken with the Hololens built-in camera is displayed in front of the user. The edges are displayed at a fixed distance from the user and follow the user when moving or changing gaze direction.

4. *X-ray with dynamic edge overlay displayed on the 3D space*. This visualization is similar to the second one, but here the edge overlay gets updated continuously using streamed camera images from the Hololens camera.

5.4.1. Simple X-ray cutout

This visualization does not provide any information about the occluding surface. It only performs the X-ray cutout to open a window on the 3D model. The user can see the objects located behind the wall through the window. The user relies on his eyes to estimate the depths of the displayed virtual objects. The Hololens provides a separate image for each eye, thus creating a stereoscopic effect, mimicking the human visual system. The depth perception from given stereo images is known as stereopsis. That is a binocular depth cue resulted from the horizontal disparity of the two images. Our vision system uses this cue to understand the depth order of the surrounding objects. This happens naturally inside our brains. The same principle is applied to sensing the depth of virtual objects using stereo images. If the images are rendered to provide a stereoscopic view, the brains are able to estimate the distance of the virtual objects displayed on the images. Depth perception does not rely only on stereopsis but uses other monocular cues as well, such as occlusion and shading. Figure 26 shows a screenshot of the visualization taken from the Hololens. We have put a cube behind the wall to serve as an occluded object.

5.4.2. X-ray with static edge overlay

A common problem with X-ray systems is that the virtual objects behind the occluding surface may appear floating in front of the user rather than behind the surface. In this case, the system fails to show occlusions correctly. This problem is solved in the literature by displaying features on the occluding surface [23] [47]. However, displaying a lot of information about the occluder may also have a negative effect and confuse the user. This is known as "superman's X-ray vision" problem [53]. In this visualization, we display features on the occluding surface using Sobel edge detection algorithm, in order to help the user to percept the depth correctly. We create an edge overlay from the image texture of the 3D model. The overlay is attached to the 3D wall and designed to show some relevant edges only. Both the X-ray and the Sobel edge detection algorithms are implemented on separate shaders. We create a material for each shader and attach it to the 3D model. The two shaders work independently. The X-ray cutout shader performs the needed cutouts to open a window on the 3D model, while the edge

Figure 26. The visualization 1 does not render any information about the occluding surface. The user relies on his eyes to estimate the depth of the virtual objects.

shader shows features from the texture. Figure 27 illustrates the attached materials and their dependencies.

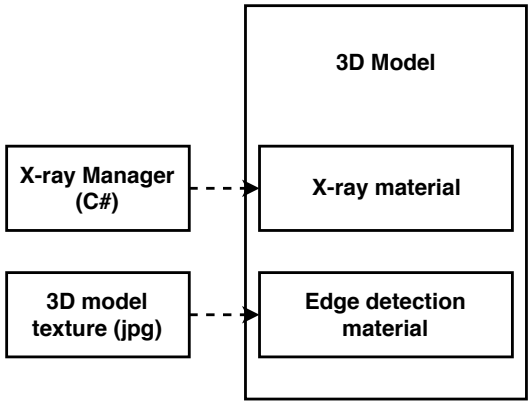

Figure 27. In visualization 2, two materials are attached to the 3D model. The X-ray material uses values given by the X-ray Manager script, while the edge detection material uses the image texture of the 3D model.

Sobel operator [67] is a classical algorithm in the field of image processing for extracting edges. It is also known as Sobel filter and Sobel edge detection. Since an edge is an indication of discontinuity in the intensity function, the edge causes a higher gradient value. The operator uses two 3x3 kernels to estimate the gradients. The Sobel operator works like a high-pass filter and gives gradients in vertical or horizontal direction only, depending on which kernel of the filter is used. To reduce noise, it uses a small window filter around the target pixel. Usually, we perform both vertical and horizontal filtering and combine the obtained results. Different kernels with different

dimensions and values can be used as well, yet the kernels associated with Sobel filter have the following values:

$$x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

$$y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

In our edge detection shader, we used the above 3x3 Sobel kernels. The algorithm takes the image texture of the model as an input and outputs an image with highlighted edges. The result is applied to the 3D model on the runtime. Notice that the input image is the same during the whole visualization. It is not updated or changed, and thus, the output remains the same. This is why the edge overlay is called static. The algorithm processes all the pixels of the input image. We compute first the intensity of the pixel of interest and the 8 neighboring pixels around it. After that, we estimate the gradients for both vertical and horizontal directions separately. We take the color values of a given pixel and its 8 neighbors and multiply then with the 3x3 kernels then we sum the result of the vertical and horizontal directions. Finally, we combine the sums into one result. Our shader has a capability to change the color of the edges as well as their intensity. The algorithm runs very smoothly. It benefits greatly from the computational power of the GPU. The same algorithm implemented with C# would run slower. Figure 28 shows a screenshot of the visualization during runtime.

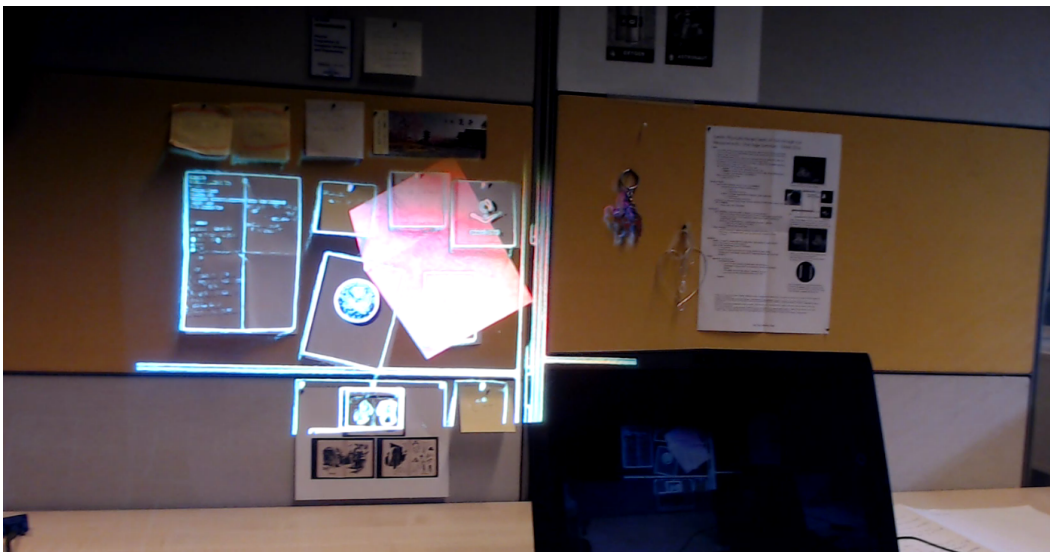

Figure 28. The visualization 2 aligns a static edge overlay on top of the occluding surface.

5.4.3. X-ray with dynamic edge overlay displayed on 2D plane

Hololens has a built-in camera mounted in front of the headset. Developers can use it to take images and videos and apply different image processing algorithms on them. In this visualization, we use live camera images taken with Hololens to generate information that may help the user in the depth estimation. We compute an edge overlay based on the images using the same Sobel edge detection algorithm discussed earlier. We display the edge overlay at a fixed distance in front of the user. If the user changes his position or rotates his head, the edge overlay follows him. This allows the user to see the edges not from the occluding wall only, but from all the real objects around him as well. We developed this visualization to address the problem associated with the depth perception. That is, the virtual objects appear to be floating in front of the blocking surface instead of appearing behind it.

In order to display the updated edge overlay for the user, our algorithm runs the following loop continuously: captures a picture of the scene, applies image processing on it, and projects the results back into the user's view. We need three essential components for the loop to work; a camera to take the pictures, a projector to project the edges, and a display panel to receive the projected edges and show them to the user. The camera is a physical device, while the projector and the display panel are purely virtual. Hololens' built-in camera supports 4 possible image resolutions. We choose the lowest one, i.e. 896x504. It is suggested by the manufacturer to use this resolution for image processing tasks. All the images have 16:9 aspect ratio. For projecting the edges, we use a Unity object called Projector. It comes with Unity's standard assets. That works like a real world projector. It allows projecting a material onto all objects facing it. The material must use a special type of shader called projection shader. The Projector has multiple parameters that can be adjusted. These include the field of view, aspect ratio, near clip plane, far clip plane, and more. We assign these values to be the same as the values of the physical camera. That way the projected edges have the same size than the real edges. For the display panel, we create a simple transparent object that has a rectangular shape. The projector is placed at the location of the camera, while the display panel is placed in front of the user. The user does not see the projector but sees the edges displayed on the panel.

We create a material for the X-ray cutout and attach it to the 3D model, and a material for the edge detection and attach it to the Projector, as shown in Figure 29. For the Projector to work correctly, a C# script called Streaming Manager is used. The script requests images from the Hololens' camera. When an image is ready, it locates the camera at the time when the image was taken, moves the projector to that position, and then transmits the image to the material to be processed by the edge detection algorithm. Obtaining the location of the camera, at the time it captured the image, is extremely important to align the virtual edges with the real ones. If this step fails, misalignments occur. Every time when Hololens captures an image, it also stores some metadata with it. That enables the calculation of the position of the camera in the world at the time when the image was taken. In our scenario, we need to know its location and rotation to place the Projector correctly. Unity API provides a 4x4 matrix called cameraToWorldMatrix at the time the photo was captured. It allows calculating the pose of the camera if the location data is available. We use it to get both the position

and orientation of the camera, to properly position the Projector. Figure 30 shows a screenshot of visualization 3.

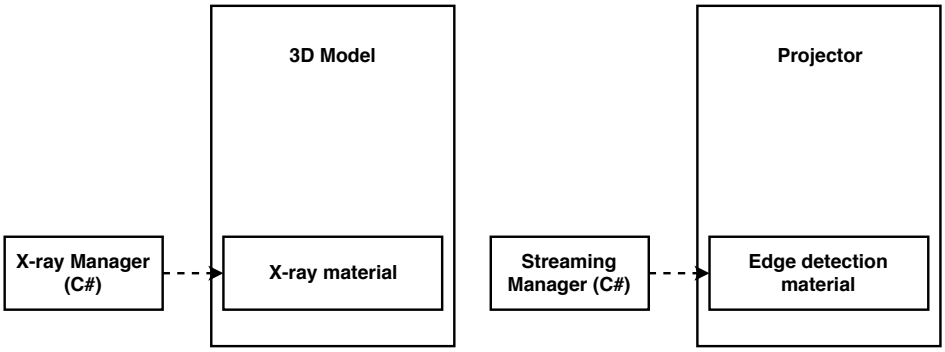

Figure 29. X-ray material is assigned to the 3D model, while the edge detection material is assigned to the Projector.

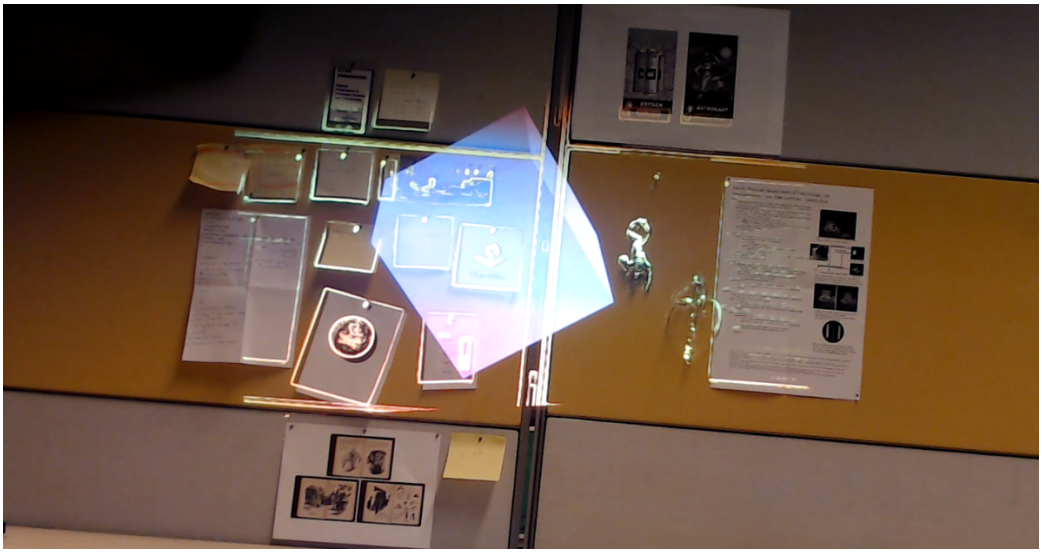

Figure 30. The visualization 3 displays an edge overlay on a 2D plane placed in front of the user.

5.4.4. X-ray with dynamic edge overlay displayed on the 3D space

In this visualization, we use live images taken with the Hololens camera to create a dynamic edge overlay. The edge overlay is aligned on top of the 3D model as in visualization 2. However, the overlay gets updated continuously using the streamed camera images. Figure 31 shows a screenshot of the visualization. We have developed an algorithm using OpenCV library to process the captured images in real-time. The algorithm is implemented as a C# script and requires a simple shader that allows updating its texture on runtime. The script reads an image from the built-in Hololens camera,

processes it and sends the output image to the shader. We call the script as OpenCV Streaming Manager and the material of the shader as Texture update material. Both materials are attached to the 3D model, as illustrated in Figure 32.

Figure 31. The visualization 4 uses a continuously updated edge overlay aligned on top of the occluding surface.

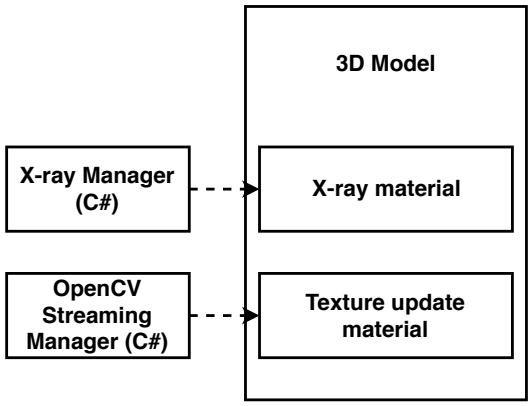

Figure 32. In visualization 4, the X-ray and texture update materials are attached to the 3D model. Each material works together with a C# script.

Our algorithm uses the original image texture of the 3D model as a reference. On runtime, it searches for similarities between the captured images and the reference image. If sufficient amount of similarities are found, the edge overlay is updated. If only a part of the wall is detected within the live image, the overlay only updates the edges of the detected part. The flow diagram of the algorithm is illustrated in Figure 33. First, we apply Sobel edge detection on both images. We store the result to be used later. Then, we use Accelerated-KAZE (AKAZE) [68] to compute local features for both images. The AKAZE feature detection is based on KAZE [69] features and

its implementation is provided in OpenCV 3. It detects local features in real-time with binary descriptors that are invariant against changes in scale and rotation. We create descriptors for the detected features in both images. Next, we apply k-nearest neighbor (kNN) feature matching, with k = 2, to match the descriptors. We use the matches to calculate the homography. The RANSAC method is applied to reject the outliers from the matches. The homography represents the relationship between the reference and a live image.§ We use the homography to transform the image resulted from the Sobel edge detection on the live image. We get the edges of the part of the wall that have been captured on the current image frame. We fill the undetected part of the wall with the edges from the reference image. This process is repeated for every image frame.

In order to optimize the algorithm, we created a separate thread for the image processing besides the main thread, as suggested in the Hololens documentation. The image processing thread runs in the background, and when a result is ready it notifies the main thread. This architecture increased the FPS of the app. The following design was used:

1. Main Thread: compute edge detection for the reference image
2. Main Thread: compute AKAZE features and descriptors for the reference image.
3. Main Thread: start the camera device
4. Main Thread: read a new image from the camera
5. Main Thread: pass the new image to the processing thread
6. Processing Thread: compute edge detection for the image
7. Processing Thread: compute AKAZE features and descriptors for the image
8. Processing Thread: match the descriptors and compute the homography matrix
9. Processing Thread: compose the final image for the edge overlay
10. Main Thread: pass the final image to the material
11. Main Thread: repeat from step 4.

In the next chapter, we evaluate the developed visualizations.

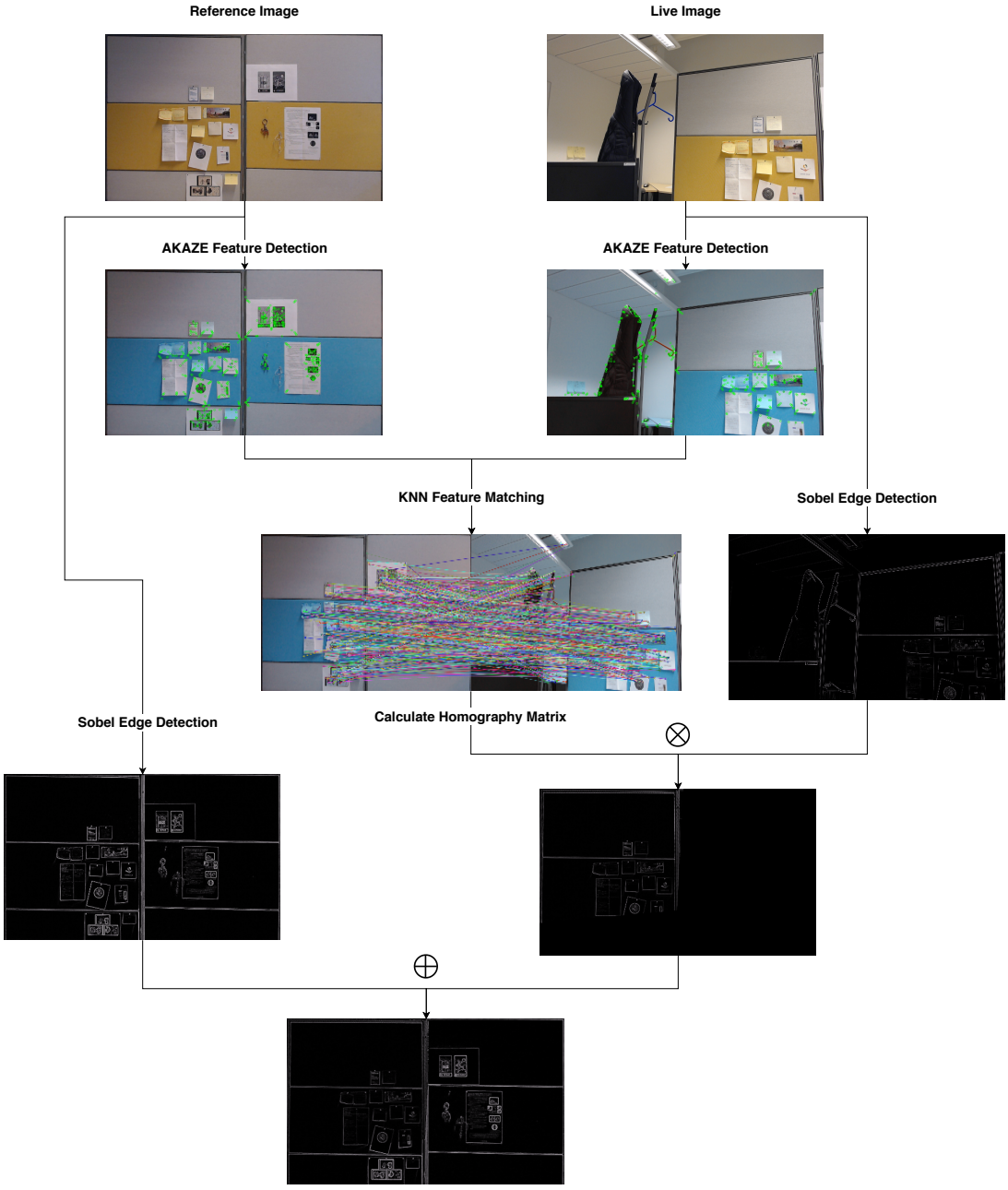

Figure 33. The different steps used on the algorithm of visualization 4.

6. EVALUATION

Every visualization, presented in the previous chapter, has a unique approach to display information about the occluding surface. This chapter presents a subjective evaluation that we conducted to test the four visualizations. The goal of this evaluation is to study the overall appearance of the visualizations. We evaluate each visualization alone to see whether it succeeds to provide the needed visual impression and depth cues. Then, we compare them against each other. The purpose of this evaluation is to find a visualization where the X-ray sight effect is the most authentic. This chapter is divided into three sections. First, we discuss the experiment design. Then, we describe the data collection method used in the experiment and finally, we analyze and study the experimental results and make conclusions.

6.1. Experiment Design

The purpose of this experiment is to study the developed visualizations and compare them against each other. Every visualization is designed to perform X-ray cutout by providing cues for depth perception. In this study, we would like to know if the techniques used on each visualization work as they should. We want to test if they deliver an impressive visualization and provide the necessary depth cues at the same time to help the users to estimate the 3D order of the objects correctly. Some visualizations display information about the occluder surface. We want to study the effects of this information to conclude whether it helps on depth perception or confuses the users. In addition, we would like to detect the errors or limitations within the implementations that were not known before. The experiment is designed to answer these questions.

In this study, we are interested in evaluating two aspects for each visualization; appearance and depth perception. Also, we want to know which visualization is the best way to illustrate the X-ray sight effect, thus provides the most impressive visualization. We have divided the experiment into two parts; a depth estimation task and a questionnaire. Both parts of the experiment are interconnected with each other. We have created a virtual cube to represent the occluded object. We divide the depth estimation task into four cycles, with each cycle having seven trials. The X-ray system uses a unique visualization on each cycle. In every trial, we change the position of the cube and ask the user to estimate the depth of the cube. We decided to use seven different positions for the cube; three in front of the wall and four behind the wall. The cube is located in a different position in each trial. Figure 34 shows the possible positions during a single cycle. The results from each cycle are recorded and studied later. The questionnaire is composed of short questions asked after each cycle, and general questions asked at the end of the experiment. The short questions have a 1-to-5 scale and are the same for all the cycles.

Overall, we have four X-ray visualizations and seven cube positions. We believe that the depth estimation task and the questionnaire are sufficient enough to evaluate the visualizations and to select the best among them. In the next sections, we present the questions asked in the questionnaire and study the results. Before conducting the real experiment, we performed a pilot test. We noticed that changing the size of the cube each time we change its position helps a lot in testing the visualization. If the

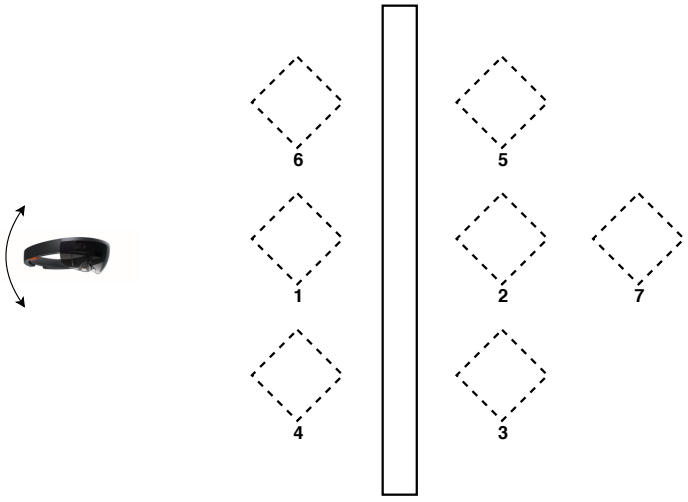

Figure 34. The possible cube positions during the experiment. Participants are asked to determine each time whether the cube is in front or behind the wall.

size of the cube is fixed, then it is easier for the users to estimate the depth just from the observing the size of the cube, but by varying the size, the users are forced to find other clues to estimate the depth. This solution tests whether the system provides the needed cues for depth estimation or not.

6.2. Data Collection

The experiment is composed of a depth estimation task and a questionnaire. For the depth estimation task, in each trial, we ask the participant to estimate the location of the cube whether it is in front or behind the wall. We write down his/her answers to analyze them later. In the questionnaire, we ask the participant a couple of questions after finishing every cycle of the experiment, and some questions at the end of the whole experiment. The questions of every cycle are in form of a Likert scale. The participants answer by giving a score in 1-to-5 scale. The score 1 means that the participant strongly disagrees with the statement, while 5 means that he/she strongly agrees with it. The questions asked at the end of the experiment are open questions. The questions asked after each cycle are listed below:

1. The system is simple to use	1	2	3	4	5
2. I feel comfortable using the system	1	2	3	4	5
3. X-ray visualization is realistic	1	2	3	4	5
4. Edges are clearly visible (*)	1	2	3	4	5

5. It is easy to locate the cube	1	2	3	4	5
6. The edge overlay helps me to see if the cube is in front or behind the wall (*)	1	2	3	4	5
7. Virtual edges are well-aligned with the real edges (*)	1	2	3	4	5
8. Holograms seem to be static	1	2	3	4	5
9. Overall I am satisfied with the system.	1	2	3	4	5

Questions 3, 6, and 7 marked with (*), were not asked for the first visualization, simple X-ray cutout, because no edges are displayed in this visualization. The questions asked at the end of the experiment are as follows:

1. Do you have previous experience with AR headsets/applications?
2. Which visualization provides the best user experience?
3. Did you experience any eye confusion or accommodation problem?
4. Can you move freely while using the system?
5. What would be good applications to x-ray visualization?
6. What improvements should we do?

In the next section, we analyze the obtained results to draw conclusions.

6.3. Experimental Results

The experiment was conducted at the Center for Machine Vision and Signal Analysis (CMVS), University of Oulu. We recruited eight voluntary participants. They are all students at the university. Four of them had experienced or developed AR applications before. Every participant wore Hololens and tested all the cycles of the experiment. The average time to complete a single experiment is around 13 minutes. We asked the participants to wear Hololens, then perform the depth estimation task and answer the questionnaire. Users can freely move inside the office and perform the experiment. For some of the participants, this was their first time to try the device. We helped them to wear the headset and explained them briefly what AR devices are capable of. We prepared also few slides about examples of augmented reality X-ray vision if the participants want to know more about the topic.

The result of the depth estimation task showed that all the participants succeeded to estimate the depth of the cube in the all visualizations, except the third visualization where three mistakes were made. The statistics are shown in Figure 35. The participants were encouraged to talk during the experiment and to express their thoughts. We recorded their answers, as well as wrote down comments and observations about the speed of answering the depth position. We noticed that the participants used more

time to answer the cube position when testing visualization 1 and 3. We can clearly see that the visualization 3 suffers from problems, which did not appear in the other ones. Even displaying the information about the occluding surface did not help to deliver the needed depth cues. We can conclude that the information displayed in the visualization 3 confuses the user instead of helping him to estimate the depth correctly.

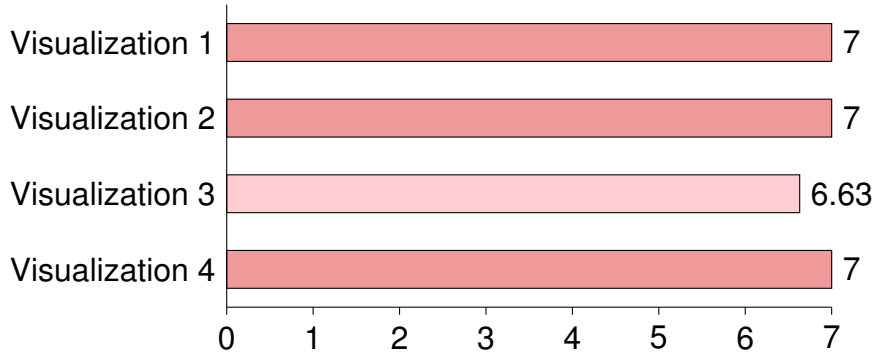

Figure 35. Average results of the depth estimation task. The total number of correct answers for each visualization is 7.

The second part of the experiment consists of a questionnaire. A series of questions are asked after trying each visualization. The total number of the questions is 9. The participant responds by giving a score from 1 to 5 to each statement. The score 1 means the participant strongly disagree with the statement, while 5 means he/she strongly agree. We compare the visualizations based on these scores. Higher scores reflect higher satisfaction. After collecting the answers, we analyzed them. We used the scores given by the participants as weights. Then we sum and average them to get the final score of each visualization. For example, if participant A answered 1, 2, 2, 4, 4, 4, 4, 5, 5 for the questions of visualization 2, we start by summing up the weights. Thus, the total weight will be 1 + 2x2 + 4x4 + 2x5 = 31. We take the average by dividing the total weight over the number of questions. We obtain 31/9=3.44. This result is the final score for visualization 2 given by participant A. We follow the same approach with the rest of the visualizations. Since we have 8 participants, we average the final scores obtained from all the participants. The final result of the questionnaire is presented in Figure 36. The result shows that the participants preferred the visualizations 2 and 4. Accordingly, the participants highlighted that the two visualizations gave better X-ray rendering, and the displayed edges helped them to find the location of the cube easily. The score of the visualization 1 is also near to the two top ones. But the score of the visualization 3 is somewhat smaller reflecting the fact that participants were less satisfied with it.

At the end of the experiment, we asked the participants explicitly what visualization do they prefer. Their answers are illustrated in Figure 37. We can see that three participants preferred the visualization 4, while the visualizations 1 and 2 got two votes. Only one participant selected the visualization 3. By comparing the overall results of the depth estimation task, the questionnaire and the questions after finishing the experiment, we can say that visualization the 2 and 4 are the best visualizations for the X-ray system. We conclude as well that the visualization 3 failed to satisfy the participants and to deliver enough depth cues. Moreover, the participants gave some

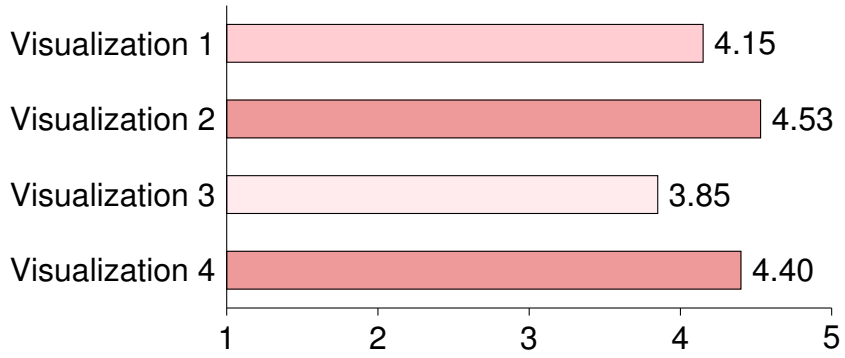

Figure 36. Average score for each visualization

suggestions as well to improve the system. Based on that, the three relevant future improvements would be 1) a better frame rate in the visualization 4, 2) use a bigger 3D model of the room and 3) better design in the visualization 3. Another asked question was to name possible applications for the developed X-ray algorithm. The participants proposed to use the X-ray system in medical applications, such as training new doctors and visualizing the internal organs. They suggested as well to use it in video games, indoor navigation, building and construction and security applications.

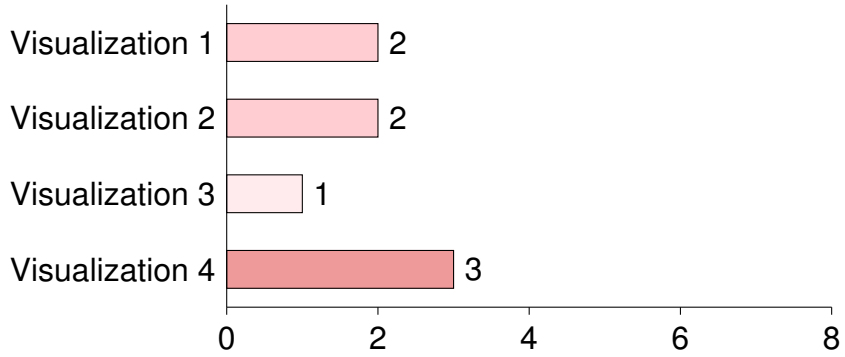

Figure 37. The answers were given by the participants to express what visualization do they like. The x-axis represents the number of participants.

7. DISCUSSION

The goal of this study was to develop an X-ray system for optical see-through head-mounted displays. So far, we implemented an X-ray vision algorithm and four rendering techniques based on it. We conducted as well a subjective experiment to test and evaluate each visualization. The results of the experiment showed that the best rendering style is to display information about the occluding surface on top of the surface itself. In our case, the visualization 2, named X-ray with static edge overlay, and visualization 4, named X-ray with dynamic edge overlay displayed on 3D space, adopt this approach. Both visualizations draw the edge overlay on top of the occluding wall. The only difference is that the visualization 4 updates its overlay continuously by extracting edges from streamed camera images taken with Hololens, while the visualization 2 computes the edge overlay only once, based on the image texture of the 3D model of the wall.

Another finding is that displaying the information of the occluding surface on a 2D plane placed in front of the user's head does not help in the depth perception. In contrast, it may lead to negative effects and confuse the user. The results of the experiments showed that the visualization 3, named X-ray with dynamic edge overlay displayed on a 2D plane, failed to provide enough depth cues for the users. Moreover, the visualization 1, simple X-ray cutout, did not display any information about the occluding wall yet outperformed the visualization 3. We can conclude that the visualization 3 mislead the user and instead of helping him in the depth perception, it made the situation more complicated.

As a future work, to improve our X-ray system further, we would like to increase the frame rate of the visualization 4. The visualization performs heavy computations, from conversion of the captured image to an OpenCV format, feature extraction and matching, homography calculation, to image transformation. This loop is bulky for the Hololens device. Finding a way to optimize it would help to increase the frame rate of the process. Another improvement is to scan the entire room instead of just part of it. That would deliver a more realistic rendering for the user. We could try one of the techniques discussed earlier in the thesis to create a 3D model of the room. Or instead of a pre-scanned 3D model, we could use the spatial mapping feature of the Hololens. That would provide untextured meshes of the environment around the Hololens on runtime. This direction has not yet been explored. The last improvement would be to add hand gestures to the X-ray system, and thus, allowing the users to interact with the system.

8. CONCLUSION

The thesis presented the development and evaluation of an augmented reality X-ray vision system for optical see-through head-mounted displays. We have developed an X-ray cutout algorithm that allows users to see through solid surfaces such as walls and facades, by augmenting the real view with virtual images representing the hidden objects. The algorithm requires a 3D model of the environment in order to represent the hidden objects. Our X-ray system is developed based on the optical see-through display, Microsoft Hololens. The algorithm is implemented on a shader file. It runs very smoothly and benefits from the high-speed computing power of the GPUs. We have developed four different visualizations as well based on the algorithm, to study the issue known as "superman's X-ray vision" problem. In X-ray vision, displaying the virtual images that represent the hidden objects without any information about the occluding surface may lead the virtual objects to appear in front of the surface rather than behind it. But showing a lot of information about the surface may confuse the user, and thus, lead to the superman's X-ray vision problem. One visualization simply renders the X-ray cutout without displaying any information about the occluding surface, while the other three visualizations present information extracted from the occluding surface. Sobel edge detection was used to extract the information. The three visualizations differ in the way how the extracted features were rendered.

The thesis presented a subjective experiment to test and evaluate the visualizations and to compare them with each other. The experiment consisted of two parts; depth estimation task and a questionnaire. In the depth estimation task, participants wore Hololens and test the X-ray system. Their main task was to estimate the depth of a virtual cube displayed sometimes in front of the occluding surface, and sometimes behind it. If the participants easily locate the cube positions, this indicates that the system provides enough depth cues so that users could correctly estimate the depth of the virtual objects on the X-ray application. If they fail, that is a sign that the system has problems with providing enough depth cues. The participants were asked to fill a questionnaire as well. The results of the experiment showed that the best rendering style is to display information about the occluding surface on top of the surface itself. However, displaying the information in a 2D panel placed in front of the user's head does not help in depth perception. In contrast, it leads to negative effects and conflicts. The results showed that the visualization that did not display any information about the occluding surface outperformed the visualization that displayed the information in a 2D panel.

There are few X-ray systems presented in the literature for optical see-through head-mounted displays. Most of the work has been presented for video see-through and hand-held displays. The work presented in this thesis tried to overcome the challenges associated with optical see-through displays and explored new rendering styles. We believe that the presented work contributes to the research community and opens new research directions to further investigate new visualization renderings and X-ray related applications.

9. REFERENCES

[1] Azuma R.T. (1997) A survey of augmented reality. Presence: Teleoperators & Virtual Environments 6, pp. 355–385.

[2] Schmalstieg D. & Hollerer T. (2016) Augmented reality: principles and practice. Addison-Wesley Professional.

[3] Milgram P. & Kishino F. (1994) A taxonomy of mixed reality visual displays. IEICE TRANSACTIONS on Information and Systems 77, pp. 1321–1329.

[4] Barfield W. (2015) Fundamentals of wearable computers and augmented reality. CRC Press.

[5] Aukstakalnis S. (2016) Practical Augmented Reality: A Guide to the Technologies, Applications, and Human Factors for AR and VR. Addison-Wesley Professional.

[6] Baricevic D., Lee C., Turk M., Hollerer T. & Bowman D.A. (2012) A hand-held ar magic lens with user-perspective rendering. In: Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on, IEEE, pp. 197–206.

[7] Hill A., Schiefer J., Wilson J., Davidson B., Gandy M. & MacIntyre B. (2011) Virtual transparency: Introducing parallax view into video see-through ar. In: Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on, IEEE, pp. 239–240.

[8] Rolland J.P. & Fuchs H. (2000) Optical versus video see-through head-mounted displays in medical visualization. Presence: Teleoperators & Virtual Environments 9, pp. 287–309.

[9] Stearns L., DeSouza V., Yin J., Findlater L. & Froehlich J.E. (2017) Augmented reality magnification for low vision users with the microsoft hololens and a finger-worn camera. In: Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility, ACM, pp. 361–362.

[10] Kijima R. & Ojika T. (1997) Transition between virtual environment and workstation environment with projective head mounted display. In: Virtual Reality Annual International Symposium, 1997., IEEE 1997, IEEE, pp. 130–137.

[11] XinReality (2017), Virtual reality. URL: https://xinreality.com/wiki/Virtual_Reality.

[12] XinReality (2017), Google glass. URL: https://xinreality.com/wiki/Google_Glass.

[13] Epson America I. (2018), Moverio bt-300. URL: https://epson.com/For-Home/Smart-Glasses/Smart-Glasses/Moverio-BT-300FPV-Smart-Glasses-%28FPV-Drone-Edition%29/p/V11H756020F.

[14] Barfield W. & Furness T.A. (1995) Virtual environments and advanced interface design. Oxford University Press on Demand.

[15] Rompapas D.C., Rovira A., Ikeda S., Plopski A., Taketomi T., Sandor C. & Kato H. (2016) Eyear: Refocusable augmented reality content through eye measurements. In: Mixed and Augmented Reality (ISMAR-Adjunct), 2016 IEEE International Symposium on, IEEE, pp. 334–335.

[16] Melzer J.E. & Moffit K. (2011) Head-mounted displays: Designing for the user.

[17] Kramida G. (2016) Resolving the vergence-accommodation conflict in head-mounted displays. IEEE transactions on visualization and computer graphics 22, pp. 1912–1931.

[18] Rompapas D.C., Oshima K., Ikeda S., Taketomi T., Yamamoto G., Sandor C. & Kato H. (2015) Eyear: Physically-based depth of field through eye measurements. In: Proc. 14th Int. Symp. on Mixed and Augmented Reality, Citeseer.

[19] Dodgson N.A. (2004) Variation and extrema of human interpupillary distance. In: Stereoscopic Displays and Virtual Reality Systems XI, vol. 5291, International Society for Optics and Photonics, vol. 5291, pp. 36–47.

[20] LaValle S.M., Yershova A., Katsev M. & Antonov M. (2014) Head tracking for the oculus rift. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, pp. 187–194.

[21] Kijima R. & Ojika T. (2002) Reflex hmd to compensate lag and correction of derivative deformation. In: Virtual Reality, 2002. Proceedings. IEEE, IEEE, pp. 172–179.

[22] Kijima R. & Miyajima K. (2016) Measurement of head mounted display's latency in rotation and side effect caused by lag compensation by simultaneous observation—an example result using oculus rift dk2. In: Virtual Reality (VR), 2016 IEEE, IEEE, pp. 203–204.

[23] Sandor C., Cunningham A., Dey A. & Mattila V.V. (2010) An augmented reality x-ray system based on visual saliency. In: Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on, IEEE, pp. 27–36.

[24] Ware C. (2012) Information visualization: perception for design. Elsevier.

[25] Kiyokawa K., Billinghurst M., Campbell B. & Woods E. (2003) An occlusion-capable optical see-through head mount display for supporting co-located collaboration. In: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society, p. 133.

[26] Klein G. & Drummond T. (2004) Sensor fusion and occlusion refinement for tablet-based ar. In: Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society, pp. 38–47.

[27] Bleser G. & Stricker D. (2009) Advanced tracking through efficient image processing and visual–inertial sensor fusion. Computers & Graphics 33, pp. 59–72.

[28] Pustka D., Huber M., Waechter C., Echtler F., Keitler P., Newman J., Schmalstieg D. & Klinker G. (2011) Automatic configuration of pervasive sensor networks for augmented reality. IEEE Pervasive Computing 10, pp. 68–79.

[29] Durrant-Whyte H.F. (1988) Sensor models and multisensor integration. The international journal of robotics research 7, pp. 97–113.

[30] Mitchell H.B. (2012) Data fusion: concepts and ideas. Springer Science & Business Media.

[31] Leonard J.J. & Durrant-Whyte H.F. (1991) Mobile robot localization by tracking geometric beacons. IEEE Transactions on robotics and Automation 7, pp. 376–382.

[32] Taketomi T., Uchiyama H. & Ikeda S. (2017) Visual slam algorithms: a survey from 2010 to 2016. IPSJ Transactions on Computer Vision and Applications 9, p. 16.

[33] Papagiannis H. (2017) Augmented Human: How Technology Is Shaping the New Reality.

[34] Google (2018), Arcore. Https://developers.google.com/ar/.

[35] Klein G. & Murray D. (2007) Parallel tracking and mapping for small ar workspaces. In: Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on, IEEE, pp. 225–234.

[36] Holloway R.L. (1997) Registration error analysis for augmented reality. Presence: Teleoperators & Virtual Environments 6, pp. 413–432.

[37] Guttag K. (2016), Ar/mr optics for combining light for a see-through display. URL: https://www.kguttag.com/2016/10/21/armr-optics-for-combining-light-for-a-see-through-display-part-1/.

[38] Odom J. (2017), What's the difference between hololens, meta 2 & magic leap? URL: https://next.reality.news/news/whats-difference-between-hololens-meta-2-magic-leap-0181804/.

[39] Kress B. & Shin M. (2013) Diffractive and holographic optics as optical combiners in head mounted displays. In: Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication, ACM, pp. 1479–1482.

[40] Kiyokawa K., Kurata Y. & Ohno H. (2000) An optical see-through display for mutual occlusion of real and virtual environments. In: Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium on, IEEE, pp. 60–67.

[41] Lincoln P., Blate A., Singh M., Whitted T., State A., Lastra A. & Fuchs H. (2016) From motion to photons in 80 microseconds: Towards minimal latency for virtual and augmented reality. IEEE transactions on visualization and computer graphics 22, pp. 1367–1376.

[42] Zhao J., Allison R.S., Vinnikov M. & Jennings S. (2017) Estimating the motion-to-photon latency in head mounted displays. In: Virtual Reality (VR), 2017 IEEE, IEEE, pp. 313–314.

[43] Microsoft (2018), Hololens' official website. URL: https://www.microsoft.com/en-us/hololens.

[44] Meta (2017), Meta 2 website. URL: http://www.metavision.com/.

[45] XinReality (2017), Meta 2. URL: https://xinreality.com/wiki/Meta_2.

[46] Coffin C. & Hollerer T. (2006) Interactive perspective cut-away views for general 3d scenes. In: 3D User Interfaces, 2006. 3DUI 2006. IEEE Symposium on, IEEE, pp. 25–28.

[47] Avery B., Sandor C. & Thomas B.H. (2009) Improving spatial perception for augmented reality x-ray vision. In: Virtual Reality Conference, 2009. VR 2009. IEEE, IEEE, pp. 79–82.

[48] Zollmann S., Kalkofen D., Mendez E. & Reitmayr G. (2010) Image-based ghostings for single layer occlusions in augmented reality. In: Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on, IEEE, pp. 19–26.

[49] Kameda Y., Takemasa T. & Ohta Y. (2004) Outdoor see-through vision utilizing surveillance cameras. In: Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society, pp. 151–160.

[50] Chen J., Granier X., Lin N. & Peng Q. (2010) On-line visualization of underground structures using context features. In: Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology, ACM, pp. 167–170.

[51] Maia L.F., Viana W. & Trinta F. (2016) A real-time x-ray mobile application using augmented reality and google street view. In: Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology, ACM, pp. 111–119.

[52] Yohan S.J., Julier S., Baillot Y., Lanzagorta M., Brown D. & Rosenblum L. (2000) Bars: Battlefield augmented reality system. In: In NATO Symposium on Information Processing Techniques for Military Systems, Citeseer.

[53] Livingston M.A., Swan J.E., Gabbard J.L., Hollerer T.H., Hix D., Julier S.J., Baillot Y. & Brown D. (2003) Resolving multiple occluded layers in augmented reality. In: Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on, IEEE, pp. 56–65.

[54] Rompapas D.C., Sorokin N., Taketomi T., Yamamoto G., Sandor C., Kato H. et al. (2014) Dynamic augmented reality x-ray on google glass. In: SIGGRAPH Asia 2014 Mobile Graphics and Interactive Applications, ACM, p. 20.

[55] Lowe D.G. (2004) Distinctive image features from scale-invariant keypoints. International journal of computer vision 60, pp. 91–110.

[56] Wu C. et al. (2011) Visualsfm: A visual structure from motion system .

[57] Wu C., Agarwal S., Curless B. & Seitz S.M. (2011) Multicore bundle adjustment. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, pp. 3057–3064.

[58] Wu C. (2013) Towards linear-time incremental structure from motion. In: 3D Vision-3DV 2013, 2013 International conference on, IEEE, pp. 127–134.

[59] Cignoni P., Callieri M., Corsini M., Dellepiane M., Ganovelli F. & Ranzuglia G. (2008) Meshlab: an open-source mesh processing tool. In: Eurographics Italian Chapter Conference, vol. 2008, vol. 2008, pp. 129–136.

[60] Matterport (2018), Matterport scenes. URL: `https://matterport.com/matterport-scenes/`.

[61] Occipital (2018), Structure sensor. URL: `https://structure.io/`.

[62] Matterport (2018), Matterport pro2 3d camera. URL: `https://matterport.com/`.

[63] Armeni I., Sener O., Zamir A.R., Jiang H., Brilakis I., Fischer M. & Savarese S. (2016) 3d semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1534–1543.

[64] Dai A., Nießner M., Zollhöfer M., Izadi S. & Theobalt C. (2017) Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. ACM Transactions on Graphics (TOG) 36, p. 24.

[65] PTC (2018), Vuforia sdk. URL: `https://vuforia.com/`.

[66] Software E. (2018), Opencv for unity. URL: `https://assetstore.unity.com/packages/tools/integration/opencv-for-unity-21088`.

[67] Sobel M.E. (1982) Asymptotic confidence intervals for indirect effects in structural equation models. Sociological methodology 13, pp. 290–312.

[68] Alcantarilla P., Nuevo J. & Bartoli A. (2013) Fast explicit diffusion for accelerated features in nonlinear scale spaces. In: Proceedings of the British Machine Vision Conference, BMVA Press.

[69] Alcantarilla P.F., Bartoli A. & Davison A.J. (2012) Kaze features. In: European Conference on Computer Vision, Springer, pp. 214–227.