Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Combining user- and device-perspective rendering for intuitive handheld AR

Bachelor's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Johannes Wilhelm

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Torsten Kuhlen

Registration date: 23.08.2022
Submission date: 09.09.2022

Zentrales Prüfungsamt/Central Examination Office

RWTHAACHEN UNIVERSITY

Eidesstattliche Versicherung

Statutory Declaration in Lieu of an Oath

___________________________ ___________________________

Name, Vorname/Last Name, First Name Matrikelnummer (freiwillige Angabe)
Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

I hereby declare in lieu of an oath that I have completed the present paper/Bachelor thesis/Master thesis* entitled

____

____

____

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

___________________________ ___________________________

Ort, Datum/City, Date Unterschrift/Signature

*Nichtzutreffendes bitte streichen

*Please delete as appropriate

Belehrung:

Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

Para. 156 StGB (German Criminal Code): False Statutory Declarations

Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence

(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.

(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

I have read and understood the above official notification:

___________________________ ___________________________

Ort, Datum/City, Date Unterschrift/Signature

Contents

List of Figures

Abstract

Augmented Reality (AR) applications for handheld devices are gaining popularity every year. Conventionally in handheld AR, an image of the real world is captured by a camera on the back of the device. Afterwards, the image is augmented with virtual information and displayed on the screen. This process, called device-perspective rendering (DPR), introduces misalignments between the displayed image and the view of the user. On the other hand, user-perspective rendering (UPR) is an alternative rendering method that eliminates this misalignment by rendering the displayed image from the perspective of the user. For this, in addition to the position of the device, the position of the user's eyes is tracked. There are advantages and disadvantages to both rendering methods, which means that for some applications neither DPR nor UPR would be the perfect choice. The aim of this thesis is to introduce new rendering methods that fill the gap between UPR and DPR by combining advantages from both sides. After analyzing the trade-off between UPR and DPR in more detail, this thesis lays out the implementation of three new methods. The implementation results in an application for iOS devices, which acts as a prototype for the new rendering methods. This prototype may be used to compare them to UPR and DPR in user studies, examples of which are presented at the end of this thesis. Therefore, the prototype will be helpful for future research and applications in the field of handheld AR.

Überblick

Augmented Reality (AR) Apps für Smartphones und Tablets werden von Jahr zu Jahr beliebter. Für diese Art von Apps wird üblicherweise ein Bild der realen Welt von einer Kamera auf der Rückseite des Geräts aufgenommen. Anschließend wird das Bild mit virtuellen Informationen erweitert und auf dem Bildschirm angezeigt. Dieser Prozess, der als Device-Perspective Rendering" (DPR) bezeichnet wird, führt zu Abweichungen zwischen dem angezeigten Bild und der Sicht des Benutzers. User-Perspective Rendering" (UPR) hingegen ist eine alternative Rendering-Methode, die diese Abweichungen beseitigt, indem das angezeigte Bild aus der Perspektive des Nutzers gerendert wird. Dazu wird neben der Position des Geräts auch die Position der Augen des Nutzers mit einberechnet. Beide Rendering-Methoden haben Vor- und Nachteile, so dass für manche Anwendungen weder DPR noch UPR die perfekte Wahl wäre. Das Ziel dieser Arbeit ist es, neue Rendering-Methoden vorzustellen, die die Lücke zwischen UPR und DPR füllen, indem sie die Vorteile beider Seiten kombinieren. Nach einer detaillierteren Analyse der Kompromisse zwischen UPR und DPR wird in dieser Arbeit die Implementierung von drei neuen Methoden dargelegt. Das Ergebnis der Implementierung ist eine App für iOS-Geräte, die als Prototyp für die neuen Rendering-Methoden dient. Dieser Prototyp kann zum Vergleich der neuen Rendering-Methoden mit UPR und DPR in Nutzerstudien verwendet werden, von denen am Ende dieser Arbeit einige beispielhaft vorgestellt werden. Auf diese Art wird der Prototyp für zukünftige Forschung im Bereich von AR Apps hilfreich sein.

Acknowledgements

First, I would like to thank my supervisor, Sebastian Hueber, for his guidance, feedback and support throughout the writing of this thesis.

I would also like to thank Prof. Dr. Jan Borchers and Prof. Dr. Torsten Kuhlen for examining this thesis.

Next, I would like to thank the Team at Café Stopover for the supreme work-environment and constant supply of Iced Americanos.

Most of all, I would like to thank my parents for providing me with the opportunity for my studies and all my friends for making it a wonderful time.

Thank you!

Conventions

Throughout this thesis we use the following conventions.

Source code is written in `typewriter-style` text.

Links to websites are marked blue and their URLs are given in the footnotes.

Attention – some words are emphasized.

Chapter 1

Introduction

The primary objective of this thesis is to lay out the trade-off between user- and device-perspective rendering for handheld augmented reality applications and to propose alternative rendering methods that aim to combine the advantages of both. This chapter introduces the fundamental concepts of augmented reality and rendering perspectives.

1.1 Augmented Reality

introduction to AR

In Augmented Reality (AR) virtual information is superimposed onto a view of the real world. In contrast to Virtual Reality, the user is not fully immersed in the virtual environment, but their perception of the real world is supplemented and enhanced by the augmentations [Azuma, 1997].

AR was first introduced by Sutherland in [1968] in the form of a head-mounted display (HMD). Today it is available in different forms, such as head-mounted, handheld and head-up displays. The combination of the real- and virtual views can be achieved with a semi-transparent screen (see-through display) or by capturing and displaying an image of the real world on an opaque screen (virtual transparency). One common form of AR used today is virtual transparency with handheld devices; an example for a head-mounted see-through display is Microsoft's HoloLens. The overarching, defining characteristics for all forms of AR were defined by Azuma in [1997]. They are: i) simultaneous view of the real world and virtual information, ii) real-time interactivity and iii) registration of the virtual information in 3D to create alignment between real and virtual objects. AR has now been adopted in different industries, with papers being published regarding application areas such as entertainment, education, navigation and medicine [Dey et al., 2018].

1.2 Different Perspectives for handheld AR

Thanks to increased computing power, more powerful GPUs on smaller scale and high-resolution displays, we have seen an increase in the number of consumer smartphones. With better tracking technologies through the combination of gyroscopic sensors, accelerometers and high-resolution cameras, handheld AR is one of the easiest available forms of AR. In the recent years, it has also seen an increase in research papers over HMDs [Dey et al., 2018].

DPR

In handheld AR, the effect of virtual transparency is usually achieved by displaying an image of the real world on the screen. This image is captured by a camera on the back of the device. To superimpose the virtual information in the correct way, the position of the device in space is tracked. Both views are then combined and displayed on screen. This way of rendering is called device-perspective rending (DPR) because the displayed image is completely defined by the position and orientation of the device. DPR creates a misalignment between the image that is displayed on screen and the view of the user (Fig. 1.1).

UPR

User-perspective rendering (UPR) is an alternative rendering method which minimizes this misalignment. UPR takes the position of the user's eyes into account and renders the displayed image from the perspective of the user. It addi-

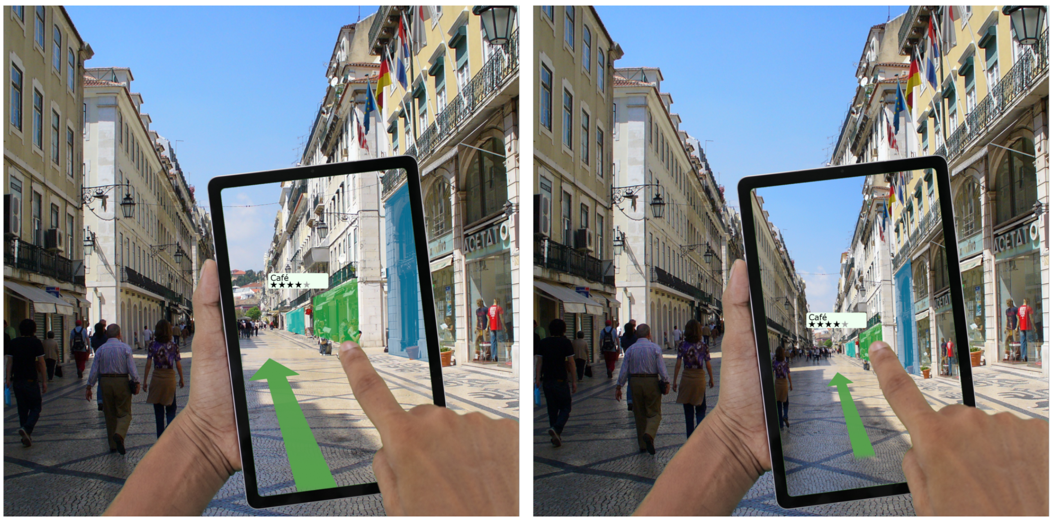

Figure 1.1: Illustration of the differences between UPR (left) and DPR (right).

tionally requires to track the position of the user's eyes and to transform the image captured by the camera. To transform the image into what the user would see if the device was transparent, a full 3D reconstruction of the scene's geometry behind the device is needed.

AUPR

Approximated user-perspective rendering (AUPR) assumes a planar depth of the scene behind the device. It reduces the amount of compute necessary while achieving a result that is similar to that of geometrically correct UPR. We will use the term '*geometrically correct* UPR' to differentiate between UPR implementations that take the full scene geometry into account and implementations that make some approximations along the way (AUPR) or limit the movement of the user (fixed-POV UPR). This enables us to use 'UPR' as an umbrella term for all rendering methods that render the displayed image from the point of view (POV) of the user.

advantages and disadvantages to DPR and UPR

There are multiple benefits of UPR over DPR. In the following chapter we will see that UPR is reported to be more intuitive to use and that it feels more natural, which leads users to prefer it over DPR [Baričević et al., 2012, Samini and Palmerius, 2016, Pucihar et al., 2013]. It was also found that users who are unexperienced in handheld AR expect the device to work in UPR [Pucihar et al., 2013]. Furthermore, UPR allows for better spatial perception [Pucihar et al., 2013], possibly because it is easier for the user to relate the content that is displayed on screen to their surroundings [Pucihar et al., 2014]. This could be especially useful in applications where the user works with augmentations that relate closely to objects in the real world (Fig. 1.1).

However, there are drawbacks to UPR too. It is computationally expensive and requires to continuously scan the 3D geometry of the scene. This makes it difficult to work in real time, especially in dynamic environments. Even when it works properly, it suffers from a significantly smaller field of view (FOV) than DPR. In DPR, the camera image, which is being augmented and displayed on screen entirely, is bigger than just the area that is seen through the device in UPR; this holds true for usual arm-length interaction distances. This effect leads UPR to perform worse in tasks where the user is looking for virtual objects in the environment [Baričević et al., 2012, Samini and Palmerius, 2016].

1.3 Thesis Outline

The trade-off between UPR and DPR is what inspired the idea for this work. Our goal was to find new, alternative rendering methods that combine the advantages of both sides. This would enable future AR applications to be more intuitive and enjoyable. We built a prototype in the form of an application for iOS, which is capable of DPR, AUPR and three new rendering methods. All three methods combine aspects of DPR and UPR and will be explained in more detail in the implementation chapter. The new rendering methods will need to be evaluated in formal user studies. In the final chapter of this work, we review user studies in the field of AR, which have been conducted on previous UPR prototypes. We propose three types of user studies and study designs that are suited to thoroughly test the new rendering methods. All three studies can be conducted using our prototype, which it is intended for. The prototype makes the different rendering methods easily accessible for future AR applications and research purposes, as it is – to the best of our knowledge – the first that does not require additional hardware but runs on off-the-shelf consumer smartphones and tablets.

Chapter 2

Related Work

This chapter is split into two parts. We will first give a summary of the most important steps in the research field of user-perspective handheld AR; from the idea of magic lenses up until the first working prototypes. We will then go over some prototypes and user studies in more detail, highlighting the impact of their results on the trade-off between UPR and DPR for handheld devices.

2.1 From the concept of Magic Lenses to handheld User-Perspective AR

magic lenses

The magic lens metaphor was first introduced by Bier et al. in [1993], who described a widget for desktop 2D user interfaces. A transparent window could be positioned above objects in an application to reveal more information about the object below. Such a see-through tool makes for an intuitive way to interact with the content around you and was later extended to 3D head-tracked environments [Viega et al., 1996, Wloka and Greenfield, 1995]. The Virtual Tricorder [1995] was a physical handheld tool, whose movements were translated to a simulated 'copy' in Virtual Reality (VR). It was able to act as a magic lens in VR, magnifying the 3D world seen through the device. The closer the user held the device to their eyes, the more of their view was affected by the lens. The metaphor of magic lenses transfers nicely to handheld AR because the device acts as a lens, which the user looks through to see an augmented view of the world behind the lens. The metaphor also illustrates why UPR can be more intuitive than DPR in such applications.

handheld AR devices

The first actual handheld AR display was built in [1995] by Rekimoto and Nagao, although interest in the field only increased later with the rise of consumer cell phones. In [2003], Wagner et al. created the first standalone handheld AR device that was not connected to a separate workstation. In 2004, the first application on a cell phone was built [Mohring et al., 2004]. Today, handheld AR is the most commonly used form of AR and next to HMDs one of the biggest research topics in the field [Dey et al., 2018]. Before the first user-perspective handheld device was built in [2011], some advances were made in UPR for video-see-through HMDs [Takagi et al., 2000, Kanbara et al., 2000, Kato and Billinghurst, 1999]. UPR is easier to solve for HMDs than for handheld devices, since the offset between the display and the user's eyes is constant and the camera(s) are relatively close to the user's eyes. In [2004], Looser et al. built a device that combined both an HMD and a handheld lens. The user's view was augmented through the HMD, but the user could alter parts of the augmentation by looking through the handheld lens.

UPR for handheld AR

The first purely handheld device capable of AUPR was built by Hill et al. in [2011]. It achieved its transparency effect by choosing the right part of the camera image and displaying it on screen. The right part is calculated depending on the relative position of the user's eyes to the device. Even though this method delivers a sufficiently looking transparency effect, it only achieves an approximation of real user-perspective. Since the camera is attached to the device and not at the actual position of the user's eyes, the image on screen is technically still rendered from the wrong perspective. The views of the user and the camera only line up at a fixed distance (or depth), which had to be chosen beforehand. To achieve geometrically correct UPR, the geometry of the scene behind the device must be calculated and the image has to be rendered from a virtual camera that is positioned at the user's eyes. The first geometrically correct UPR magic lens was built a year later by Baričević et al. in [2012]. Even though this approach is technically more correct, it comes with some drawbacks like higher hardware requirements and relatively many visual artifacts.

In the following years, various user studies were performed on different prototypes. We will go over the most relevant ones for our approach in the next section.

2.2 Existing UPR Prototypes and related User Studies

UPR is more intuitive but it suffers from a small FOV

Baričević et al. performed a simulation-based study in [2012]. Their goal was to compare user-perspective magic lenses to device-perspective magic lenses in different sizes. The study was performed in VR , as the current technology did not yet allow for a fair comparison on actual prototypes. The participants had to perform a task that was split into two phases: searching a virtual target with the magic lens and then touching it with their hand behind the device. The results of the study indicated that the users generally preferred the UPR lens over the DPR lens. The performance of the UPR lens was however a little slower on smaller display sizes, especially during the search phase of the task. This makes sense due to the smaller FOV of UPR lenses. The authors concluded that it would be reasonable to use UPR lenses in small work environments and DPR lenses in outdoor environments, for faster searching.

These results were later replicated outside VR on a physical prototype by Samini et al. in [2016]. Using an external tracking system and cameras attached to a screen, they were able to build a device which approximated user-perspective rendering. They conducted multiple user studies with this setup, including a search and select task and different object manipulation tasks. After the search and select tasks, participants reported that the bigger FOV of the DPR lens made it easier to find objects, but that the UPR lens was more intuitive to use.

The reason we focus so much on selection tasks is that it is a common action that almost always precedes other actions a user may perform. For a more general overview and comparison of different selection techniques for AR we refer to [Looser et al., 2007].

UPR enables better spatial perception but its benefits are limited

Pucihar et al. [2013] confirmed in their own experiments, that UPR is preferred over DPR in some tasks. Furthermore, they found out that users initially expect AR to work from the user's perspective and that the dual-view problem created by DPR distorts the user's spatial perception . They later extended their analysis to determine differences between UPR and DPR on the use of surrounding context [2014]. The misalignment between what is displayed on screen and the real world is a well-known issue in handheld AR, which affects the user's reliance on peripheral vision [Kruijff et al., 2010]. UPR should make it easier to relate on-screen content to what is seen outside the screen borders. While Pucihar et al. found that UPR makes cross-context interaction (tasks requiring crossing the borders of the magic lens) easier, they also noticed that this effect is limited by the diplopia and depth-of-field problems [Pucihar et al., 2014]. Diplopia – or double vision – occurs because the user's eyes can only converge at either the distance of the magic lens or the background. This results in seeing one or the other in double vision and makes it harder to align the magic lens with the background. Additionally, the distance between the nearest and furthest object that can simultaneously appear sharp to the human eye is limited. This commonly results in the background appearing blurry while the user is focused on the magic lens.

limitations to UPR

It is worth highlighting that all studies mentioned above were performed either in simulations or on prototypes that only approximate true UPR. The first geometrically correct UPR prototype by Baričević et al. [2012], was implemented using the help of a Kinect-Sensor and a Wii-Remote for somewhat accurate tracking. Since then, improvements have been made on these devices. Mainly Baričević et al. improved on their first prototypes by using stereo matching algorithms and image based rendering [2014, 2016]. Other approaches were taken as well, for example to achieve geometrically correct UPR by segmenting the camera image into regions of different depths [Kyriazakos and Moustakas, 2015]. But in contrast to AUPR implementations that can run on stand-alone devices [Hill et al., 2011], all these implementations need a separate workstation for the relatively high computational demand. Furthermore, no user studies were performed on these prototypes. This makes it difficult to estimate the benefit of geometrically correct UPR over AUPR, which would be especially interesting for common interaction tasks in AR applications. With multiple studies indicating that AUPR already yields most – if not all – of the benefits one would expect geometrically correct UPR to have, it begs the question whether true UPR is the most useful approach for some applications. Going even further, we have seen that DPR has multiple advantages over UPR, such as a bigger FOV. Since the diplopia and depth-of-field problems naturally limit the benefit of perfect alignment between on-screen content and the real world, it could be useful to explore alternative rendering methods between UPR and DPR. In the following we will go over the implementation of a prototype that enables the comparison between AUPR, DPR and new rendering methods, which aim to combine the advantages of UPR and DPR.

Chapter 3

Building a Prototype for UPR-based Rendering Methods

In the previous chapter we gave an overview of past developments in the field of handheld AR and user-perspective rendering. Next, we will go over the process of building our own prototype. We will start with our ideas and goals; we will go over the hardware and software we used and then cover our implementation of AUPR itself. Afterwards, we will explain how our new rendering methods are implemented and how they differ from UPR and DPR. Lastly, we will go over some observations we made while implementing and testing the prototype.

3.1 Motivation and Intentions

As seen in the previous chapter, there exists a trade-off between UPR and DPR. The benefits of UPR include: (i) better spatial perception [Pucihar et al., 2013], (ii) a more natural feel, which leads users to prefer UPR over DPR [Baričević et al., 2012, Samini and Palmerius, 2016, Pucihar et al., 2013] and (iii) it makes it easier for the user to relate the surrounding context to what is displayed on screen [Pucihar et al., 2014]. DPR on the other hand, benefits from a larger FOV, which is more convenient for searching the scene for virtual objects [Baričević et al., 2012, Samini and Palmerius, 2016] and could also help when looking at large virtual objects. Furthermore, it runs more stable due to lower hardware requirements, which makes it more suitable for current AR applications. To find a solution to the trade-off between the two – and to maximize usability and enjoyment for future AR applications – we built a prototype with new rendering methods that aim to combine the advantages of both sides.

idea of new rendering methods

The difference between what is displayed on screen in UPR and DPR, originates from the different camera frustums. The two main differences are the position of the camera frustum and its rotation and shape. This makes it intuitive to approach the problem of finding new rendering methods from two sides. It is possible to start with the camera position from either UPR or DPR and then change the shape (and/or rotation) of the frustum to approach that of the other rendering method. Of course, it is also possible to alter both position and shape. This way we end up with three new rendering methods. One method renders the scene from the position of the user's eyes with an increased FOV ($M1_{FOV}$). The second method takes the position and shape of the frustum from DPR, but responds to changes in the user's head position by rotating the frustum ($M2_{Rot}$). The third method changes both position and shape of the frustum by interpolating directly between UPR and DPR ($M3_{Int}$). We will cover the implementation of these methods in more detail in the following section.

With the new rendering methods, the alignment between on-screen content and surroundings will not be perfect. This is a sacrifice we are willing to make, because as seen in the previous chapter, the benefit of perfect alignment is limited by the diplopia and depth-of-field effects anyways. We think that the combination of different advantages, like the displayed image responding to changes in the user's head position and a bigger FOV, can outweigh the disadvantage of slightly worse alignments. This will later need to be validated through user studies. To enable such studies and to compare the new rendering methods to DPR and AUPR, we built a prototype in the form of an application for iOS.

The following section will cover the implementation of this prototype in full detail.

3.2 Implementation

AUPR as a base for the new rendering methods

We first implemented our own version of AUPR , expanding on the ideas of [Samini and Palmerius, 2014]. This was useful because our new rendering methods build upon the concepts of UPR and our AUPR implementation functions as a base for them. It also makes it possible to later compare the new methods to the original AUPR method. It is worth noting that we chose to not build up from geometrically correct UPR, because any minor benefits of the better alignment would later be lost in the new methods anyways. Furthermore, it is expected that in most applications the user will be focused on a specific area of interest at a certain depth, and small misalignments at other depths will be hardly noticeable. Understanding our implementation of AUPR makes it easy to understand the new rendering methods. Therefore, in the following, we will first go over our implementation of AUPR and explain how the new methods differ from it afterwards.

geometric approach to our AUPR implementation

We are going for a general setup as illustrated in Fig. 3.1, which is inspired by the geometric approach in [Samini and Palmerius, 2014]. Some implementation details are different, and we improved upon it by making it work more accurately at small distances. With this setup, everything that is needed for AUPR can be split into four key components: (i) face and device tracking, (ii) calculating the camera frustum, (iii) positioning the image plane and (iv) measuring the scene depth. After briefly going over the hardware and software we used, we will explain each of these components in detail.

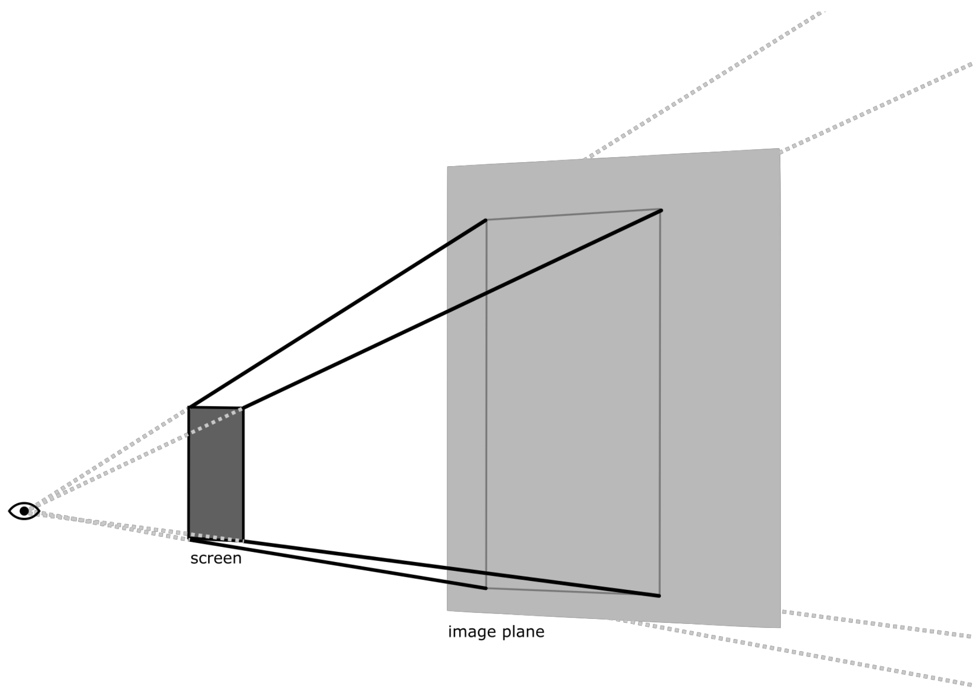

Figure 3.1: The main components of our AUPR implementation are (a) the screen of the device, (b) the image plane depicting the most recent image captured by the device's camera and (c) the frustum of the virtual camera , originating from the position of the user's eyes.

3.2.1 Hardware and Software used

the prototype was built in Unity, for iOS devices

The application was built in Unity and was tested on both an iPhone 12 Pro and an iPad mini 6. The iPhone comes with 6 GB of RAM and has multiple cameras, including a back facing ultrawide lens and a front facing TrueDepth camera. Other relevant sensors include the back facing LiDAR sensor – which is the main reason why we chose it –, a gyroscope and an accelerometer. The iPad in comparison, comes with 4GB of RAM, has no LiDAR sensor and has the ultrawide camera on the front. These differences make it a nice addition to the iPhone for testing. On both devices, the back-facing cameras used for our application have an approximated FOV of 60 degrees. No further tracking hardware was used, which – to the best of our knowledge – makes our prototype the first working handheld AUPR implementation to work on standalone consumer smartphones and tablets.

The application itself was built using the game engine Unity and its AR Foundation framework . AR Foundation is a multi-platform interface that builds on top of different AR frameworks, such as Apple's ARKit for iOS and ARCore for Android. AR Foundation provides useful functionality from the underlying ARKit framework, such as plane detection and access to the LiDAR sensor. Furthermore, Unity provides a 3D work environment, making it easy to work in different coordinate systems and with virtual cameras.

3.2.2 Face and Device Tracking

For every AR application it is necessary to be able to track the real-world environment and the position of the device within it. For that purpose, Apple's ARKit uses a combination of motion sensor data and visual landmarks in the camera video feed [AppleDevDoc/WorldTracking[1]]; a technique called visual-inertial odometry. The origin of the world-space coordinate system will be located at the position of the device's camera when the app is opened. From there on, the device's position and orientation will continually be updated. For AUPR it is also necessary to know the position of the user's eyes relative to the device. AR Foundation can use the iPhone's front-facing camera for face tracking and can even recognize facial expressions. This way the positions of the user's eyes are determined and whether they are open or closed. The midpoint between the user's eyes is calculated or – if one eye is closed – the position of the open eye is used. This tracked position is then smoothed by averaging it over a window of the last few known positions. Once the current positions of the user's eyes and the device are known, this information is used to update the location and frustum of a virtual camera. This virtual camera is located at the user's eyes and renders the view of the scene that will be displayed on screen.

the positions of the device and the user's eyes are tracked and continually updated

[1]developer.apple.com/documentation/arkit/configuration_objects/understanding_world_tracking

3.2.3 Calculating the Camera Frustum

the virtual camera frustum is defined by the positions of the screen corners and the user's eyes

The opaque device covers up part of the user's view. This hidden part must be displayed on screen, to achieve the transparency effect AUPR is aiming for. For this, the frustum corners of the virtual camera need to be cast from the user's eyes through the corners of the screen. The coordinates of the screen corners are dependent on the device's dimensions, which are measured by the distance of its rear camera from its screen borders. From the position, orientation and dimensions of the device, the positions of the screen corners are calculated. The frustum of the virtual camera is defined by its projection matrix and will take the shape of a truncated pyramid. It is limited in the front and back by the near and far clipping planes, which can be chosen more-or-less arbitrarily. Everything that is inside this frustum will be displayed on screen.

near clipping plane and projection matrix

With the necessary information provided, the projection matrix of the virtual camera is calculated. The projection matrix is responsible for mapping 3D points of the virtual world to 2D points on the image that will be displayed on screen. Every point within the frustum will be projected onto the near clipping plane. The projection matrix can be built from six parameters: the z coordinates of the near and far clipping planes (n and f) and the x or y coordinates of the four sides of the near clipping plane (l, r, t, b). All coordinates must be given in relative position to the virtual camera (Fig. 3.2). The projection matrix takes the general shape of:

$$\begin{pmatrix} x & 0 & a & 0 \\ 0 & y & b & 0 \\ 0 & 0 & c & d \\ 0 & 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

where x, y and c are scaling factors for the x, y and z coordinates; d is a constant offset for the z value and a and b are linear offsets for the x and y values (multiplied by z). We refer to OpenGL Projection Matrix by Song Ho Ahn[2] for a detailed derivation of the matrix entries.

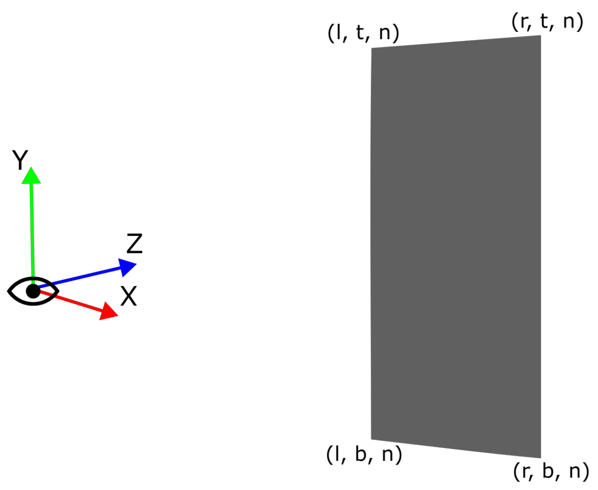

Figure 3.2: Measurements of the near clipping plane. They are used to calculate the projection matrix of the virtual camera.

In our case it is fitting to use the device's screen as the near clipping plane. This leads the values l, r, t and b to be directly equivalent to the screen's dimensions (Fig. 3.2). The frustum now takes the shape as depicted in Fig. 3.1 and will update whenever the user moves their head or the device. Alternatively, instead of setting n to be equal to the distance d_{screen} to the screen, it is possible to choose a smaller value and multiply the other values by n/d_{screen}. This way, virtual objects in front of (or peeking out of) the screen will be visible as well.

With the camera frustum set, all virtual objects placed behind the device will be rendered from the user's perspective. The screen will act as a window into the virtual world (Fig. 3.3).

[2]songho.ca/opengl/gl_projectionmatrix.html

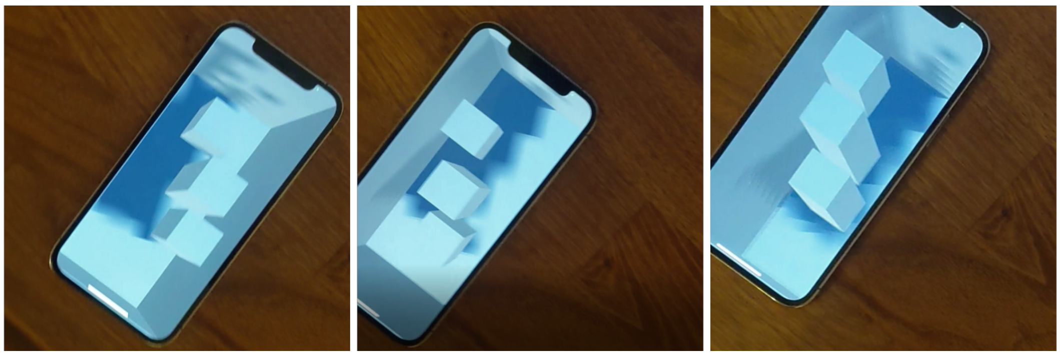

Figure 3.3: A depth illusion effect created by rendering the virtual scene from the perspective of the user.

To achieve the desired transparency effect however, a part of the image that is captured by the device's camera needs to be displayed on screen. For this, a virtual plane that is textured with the latest camera image is added to the scene. This image plane is placed behind the device in the virtual world and will be partly inside the created camera frustum.

3.2.4 Positioning the Image Plane

Positioning the image plane correctly is essential to achieving the desired transparency effect. Its relative position and orientation to the device will directly influence which part of the image is inside the frustum. For the desired transparency effect, objects on the screen must appear to the user in the same sizes and positions as they do in the real world.

the image plane is positioned in way that the on-screen content lines up with the surrounding view

As in [Samini and Palmerius, 2014], the image plane is positioned straight behind the device's camera and acts as a two-dimensional slice of its frustum in the virtual world (Fig. 3.4). The size of the image plane can be chosen arbitrarily, as a larger size balances out with a greater distance. It is however important that the image plane covers the width of the frustum of the device's camera. With its size set to 10x13.33 meters, the distance from the device's camera is determined by the camera intrinsics or FOV (roughly 10m for the devices we tested on, which is equivalent to a FOV of approximately 60 degrees).

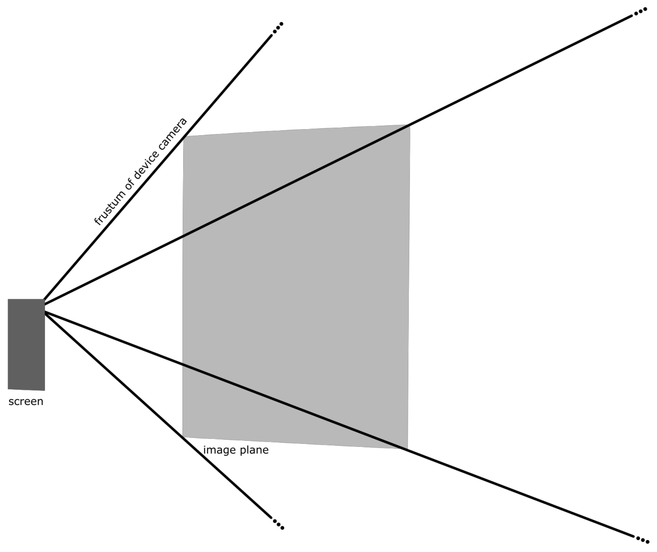

Figure 3.4: The size and position of the image plane is chosen in a way that it acts as a slice of the frustum of the device's camera.

Figure 3.5 illustrates the reasoning behind this approach: objects seen through the device appear approximately in the same position on the screen as they do in the real world and their position is stable when tilting the screen. This is the behavior one would expect from looking through a transparent sheet of glass. This approach works as long as the distance between the user's eyes and the device is negligible compared to the distance between the device and the observed object. We will discuss what happens in the other case after the following paragraph.

correcting misalignments

First, another issue is that the sizes in which objects appear on screen is dependent on their distance from the device's camera. This too becomes a problem when the distance between the user's eyes and the screen becomes large in comparison to that between the device and the observed object. In small environments, where the observed scene is less than two meters away from the device, objects tend to appear too big on screen. This would later result in virtual objects appearing too small in comparison to their background. To counteract this problem, the distance between image plane and device is increased when the depth

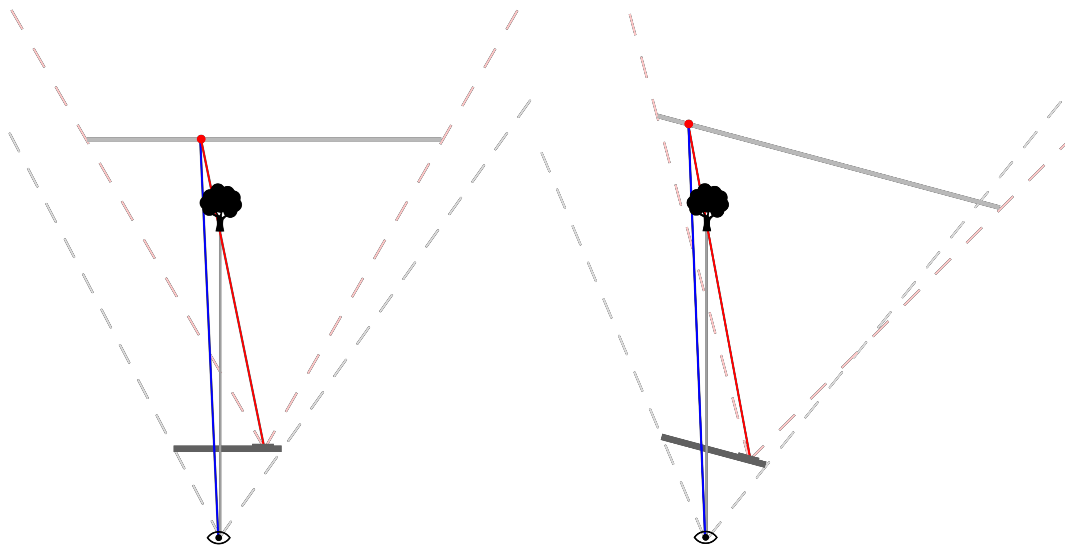

Figure 3.5: The correct positioning of the image plane results in an approximately correct alignment between the on-screen content (blue line) and the view of the user (grey line).

the image plane is moved back so that real-world objects appear in the correct sizes

of the scene becomes too small. This way, the content on the image plane will appear smaller to the virtual camera. The depth of the scene is measured using the LiDAR sensor. The function that relates the depth of the scene to the distance of the image plane was obtained through empirical testing and is shown in figure 3.6. The steepness of this function should to some degree be dependent on the intrinsics of the camera and even the size of the screen. Some error is however expected, as the image on the image plane will always be captured from the perspective of the device camera and not from the actual perspective of the user. Also, other measurements such as the depth of the scene are only estimates themselves. Therefore, it is accurate enough to estimate this function through empirical testing. We found that it even generalizes well to the iPad mini, which has an entirely different camera, screen size and aspect ratio.

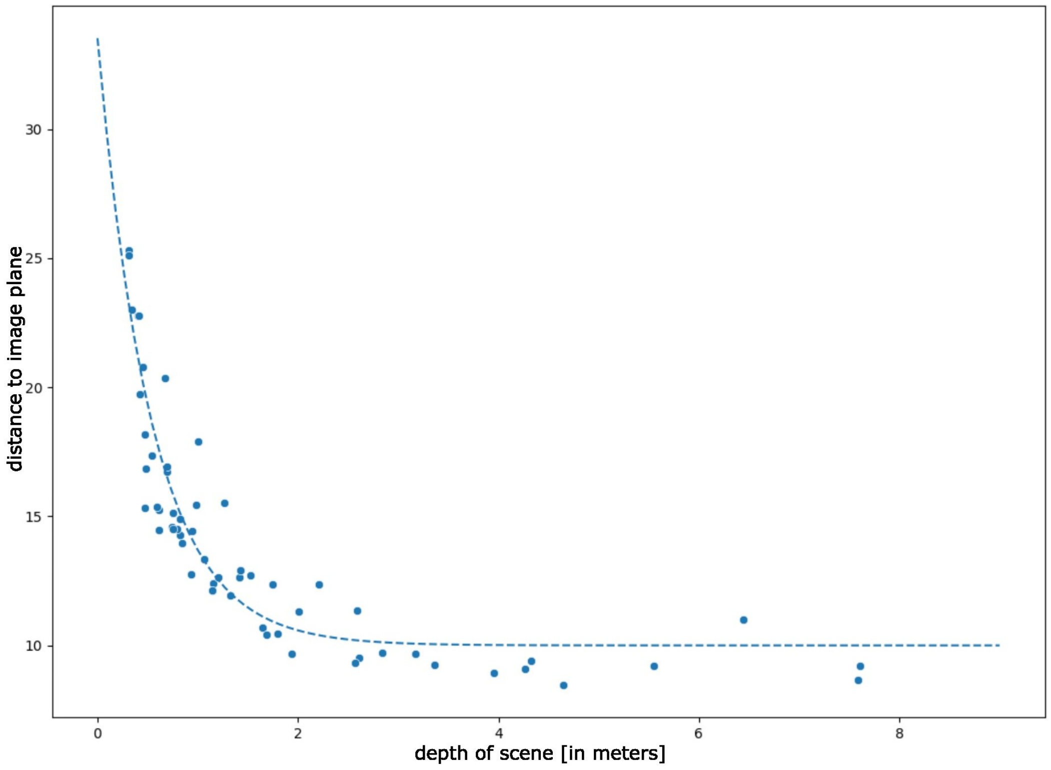

Figure 3.6: The closer the objects in the real world are to the device, the further away the image plane has to be placed in the virtual world. The relation between the two is depicted in this function: $\text{distance} = 10 + 23.5e^{-1.85\,\text{depth}}$

By just moving the image plane back however, a new problem is introduced. The image plane now became too small to cover the FOV of the device's camera. This leads to further misalignments when looking at the screen from the side (Fig. 3.7). To keep the right part of the image displayed on screen, the image plane is rotated around the position of the device's camera. The amount of rotation is dependent on both the angle at which the user looks at the screen and the distance to the image plane (Fig. 3.8). Following the sight of the user through the center of the screen, a point p_{target} is determined at the distance of the image plane. If the image plane covered the entire width of the camera frustum, the point p_{target} would refer to the point p on the image plane. To adjust the position of the image plane, it is rotated around the device's camera until p lies on the line of p_{target}. This rotation is given through the following equations:

moving the image plane back introduces new misalignments

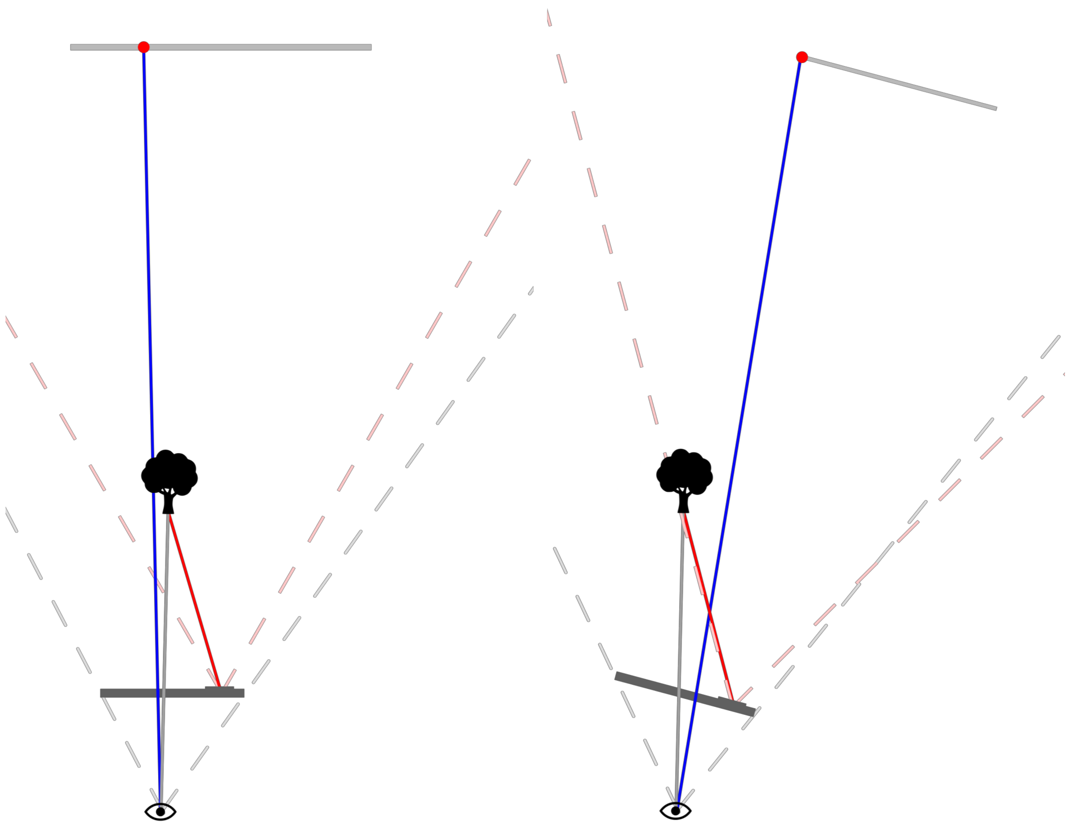

Figure 3.7: Moving the image plane back results in misalignments when looking at the screen from the side.

$$\text{rot}_{\text{horiz}} = \alpha - \tan^{-1}\left(\frac{0.5\,width}{distance}\,\frac{\tan(\alpha)}{\tan(27°)}\right)$$

$$\text{rot}_{\text{vert}} = \beta - \tan^{-1}\left(\frac{0.5\,height}{distance}\,\frac{\tan(\beta)}{\tan(34°)}\right)$$

where $width$, $height$ and $distance$ refer to the image plane and α and β are angles between the screen normal and the line of sight of the user. $27°$ and $34°$ are half the FOV of the device's camera in either direction.

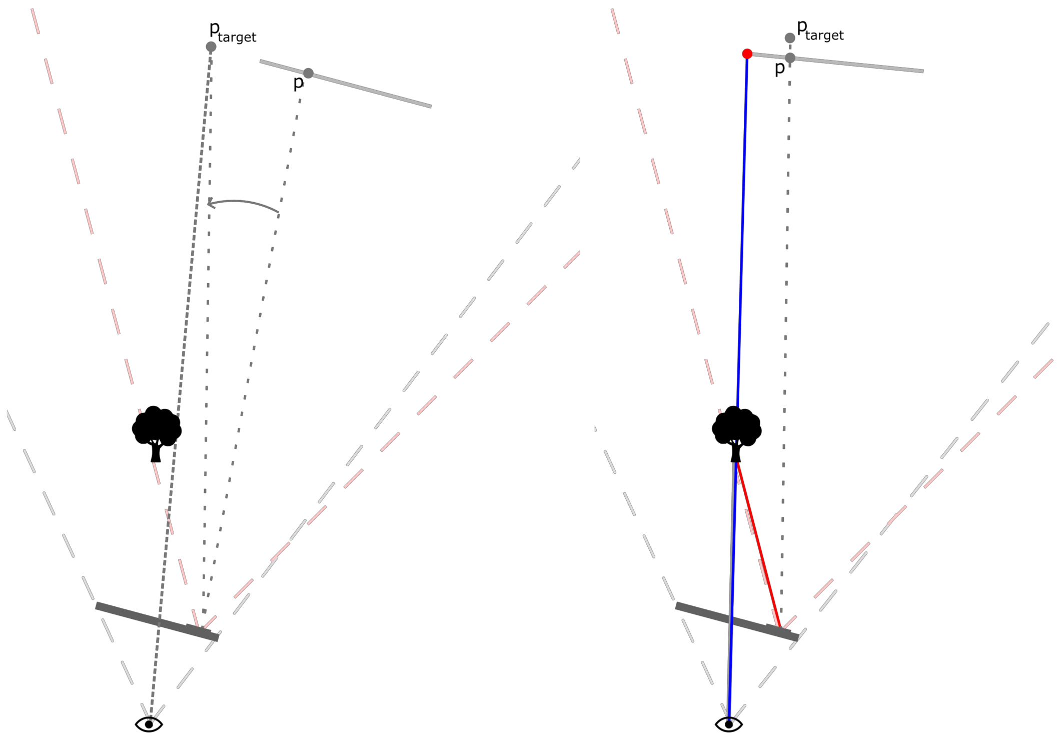

Figure 3.8: To restore the alignment between on-screen content and real-world objects, the image plane is rotated around the position of the device's camera.

With this improvement, the on-screen content now appears in the same size and position as the user would see it in the real world. The alignment now works properly up until a scene depth of ∼25cm, a significant improvement over the previous 1.5 or 2 meters.

3.2.5 Depth Measurement and Virtual Objects

Measuring the depth of the scene behind the device is needed for the correct placement of the image plane. For this, a virtual mesh of the surroundings is built using the LiDAR scanner . Rays are then cast from the position of the user's eyes through the screen. Their hit points with the LiDAR mesh are used to calculate an average depth of the scene. If no LiDAR scanner is available, standard AR

the depth of the scene is estimated using the LiDAR scanner and virtual objects can be placed within it planes can be used. AR planes are placed over flat surfaces, which can be detected without a LiDAR scanner; it makes the depth estimation slightly less accurate.

Virtual objects can be placed in the virtual world between the device and the image plane. They are then rendered by the virtual camera from the user's perspective. When placing virtual objects on surfaces in the real world, they can be marked as 'anchored'. The known position of anchored objects can then be used to estimate the depth of the scene more accurately; assuming the user's focus lies on the virtual object when it is on the screen. Virtual objects can get occluded by static real-world objects by hiding them behind the LiDAR mesh. Virtual objects also cast shadows onto AR planes, which further adds to the interconnection between the real- and virtual world.

This is all the functionality that is needed for AUPR to work properly and for the new methods to be build on top of.

3.2.6 New Rendering Methods

In addition to standard DPR and the AUPR method explained above, we implemented three new rendering methods. These methods differ from our AUPR implementation in the position of the virtual camera and the shape and orientation of its frustum.

new method 1

The first method ($M1_{FOV}$) renders the scene from the user's POV with an increased FOV . This is achieved by artificially increasing the size of the screen, moving its corners outwards (Fig. 3.9). The on-screen content stays fully responsive to changes in the position of the user's head and objects still appear stable. Additionally, the bigger FOV allows for faster searching, while the misalignment between on-screen content and surroundings is relatively small.

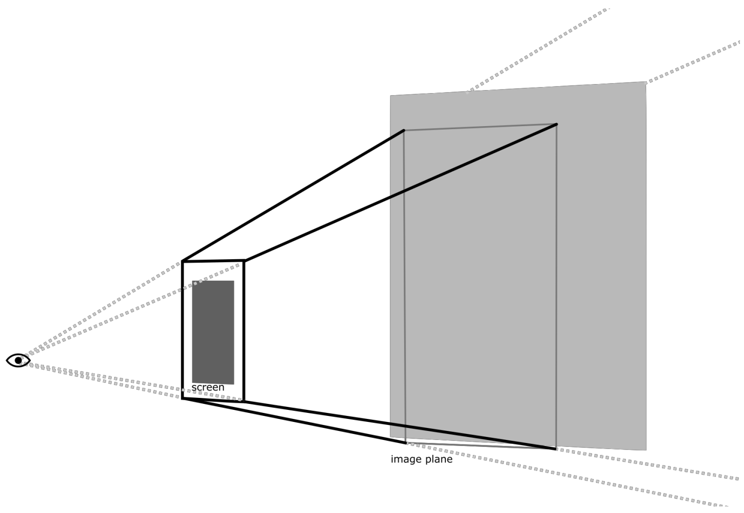

Figure 3.9: New rendering method $M1_{FOV}$: The FOV is increased by simulating a larger display size and increasing the size of the near clipping plane.

new method 2

The second method ($M2_{Rot}$) renders the scene from the position of the device's camera but responds to changes in the user's head position by rotating its frustum (Fig. 3.10). The frustum corners are defined by points on the image plane, near its corners. This method benefits from the biggest FOV, which is the same as in DPR . While the camera image and the image plane are in form factor 4:3, the screen however is narrower. Therefore, not the whole camera image can be displayed at once. By rotating the camera frustum when the user is looking at the screen from an angle, these outer parts of the captured image become visible; they would normally go unused.

new method 3

The third method ($M3_{Int}$) interpolates between the POVs and frustum shapes of UPR and DPR. This way of merging the two rendering methods might be the first that comes to mind. When interpolating between them, the position of the virtual camera is placed on a line between the user's eyes and the device's camera, while the frustum corners are moving from the screen corners to the corner points on the image plane (Fig. 3.11).

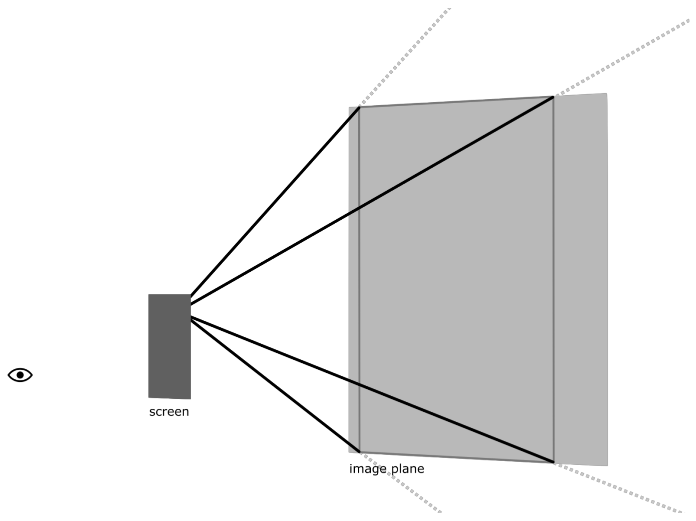

Figure 3.10: New rendering method M2$_{\mathrm{Rot}}$: The frustum of the virtual camera replicates the camera of the device but responds to changes in the position of the user by rotating.

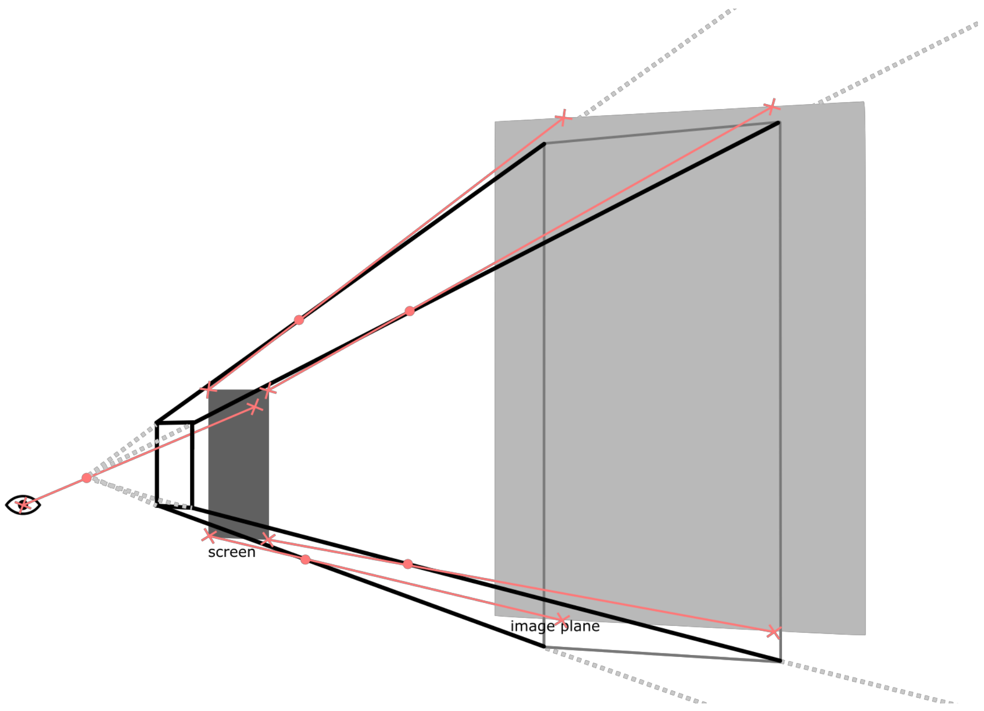

Figure 3.11: New rendering method M3$_{\mathrm{Int}}$: The frustum corners and its origin are interpolated on lines between (a) the user's eyes and the device's camera and (b) the screen corners and corner points on the image plane.

3.3 Observations from Testing the Prototype

To conclude this chapter, we will go over some of the key observations we made while working on the implementation of this protype and testing it afterwards.

on the results of our implementation

Figure 3.12 depicts results of our AUPR implementation. The alignment between the displayed image and the real-world surroundings makes for a convincing see-through effect. Edges of objects outside the screen borders continue without great distortion inside the boarders on the captured camera image. Even though our implementation only approximates true UPR by assuming a planar depth of the scene, the dynamic measuring and adjusting of this depth makes it barely noticeable. Not only is the alignment between camera image and surroundings stable, but also virtual objects that are added to the scene are anchored reasonably well to the camera image background. Both types of alignments work at large distances to the observed scene of over ten meters and at close ranges of under two meters, down to 25 centimeters; the alignments between virtual objects and their background tend to break first. This stability at close ranges is essential for AR applications in the gaming and productivity/work categories, where the user is seated in a small space and looks at a scene with shallow depth.

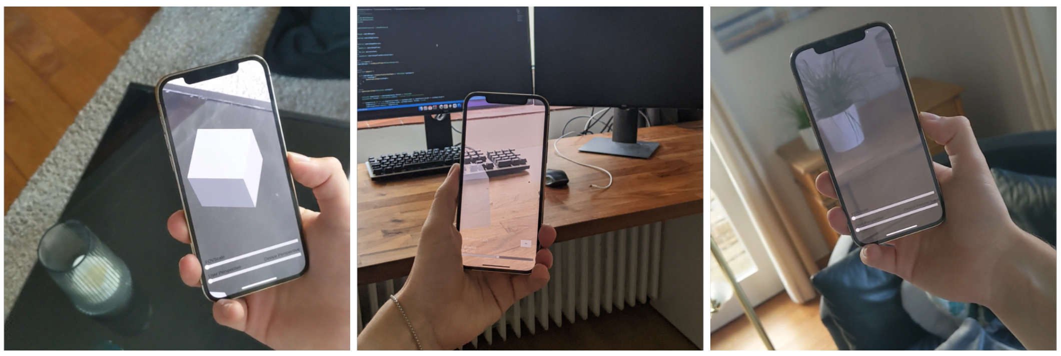
Figure 3.12: Results of our AUPR implementation.

Since the AUPR implantation functions as a base for the new rendering methods, they work within similar bounds. Keeping the position of the virtual camera fixed to either the user´s POV or the device's camera makes rendering methods $M1_{FOV}$ and $M2_{Rot}$ feel familiar because the user is used to seeing content from these POVs. The third method however, in which the camera´s origin is placed somewhere between the user´s eyes and the device´s camera, can feel less natural and sometimes has a dizzying effect. We think this is the case because there is no fixed point in space which the user can associate with the origin of the rendered image.

3.3.1 Testing the Prototype with a small group of people

user feedback

After its implementation we informally evaluated the prototype with a small group of people, ranging in age between 20 and 50 years old, with a median of 26. We observed that inexperienced users were able to intuitively understand the concept of UPR after testing our AUPR implementation. It took the users some more experimentation to get familiar with- and fully understand the new rendering methods. They were able to use sliders to manipulate the FOV in the first method or the degree of interpolation between UPR and DPR in the third. This made it easy for them to play around with the prototype and see the rendered image gradually change between different rendering methods. After some experimentation, our users preferred positions greater than one for the FOV slider, which indicates that they may prefer the bigger FOV over traditional UPR. Generally, we observed that after some time of using the UPR-based methods one gets used to seeing the world directly through the frame of the device. This goes so far that after switching back to normal DPR, it can feel weird and artificial. These observations reflect the findings of various studies which concluded that UPR tends to feel more natural to users than DPR [Baričević et al., 2012, Samini and Palmerius, 2016, Pucihar et al., 2013] and suggest that this might also hold true for our new rendering methods.

3.3.2 Potential Use Cases

We can imagine the new UPR-based methods to be useful in scenarios where the user is seated in a fixed place but still wants to look at virtual objects from different angles. In this scenario a comfortable way for the user to do so is to move their head around. The UPR-based methods make use of these changes in the position of the user's eyes and render the observed object from a different angle.

When looking at large virtual objects, it is possible to dynamically increase or decrease the FOV to fit the observed virtual object into frame; while the object is small enough or out of sight, one could return to normal UPR to get the best alignment. Alternatively, one could set a fixed FOV or let the user control it manually. The ability to manually control the FOV could be useful when interacting with virtual objects at different scales. For example, when looking at 3D renders of complicated machine parts, where the user frequently changes between looking at the system as a whole and looking at a single part in more detail. The second new rendering method $M2_{Rot}$ is just a small improvement over normal DPR, in the sense that it makes use of the otherwise wasted parts of the camera image. This should come without any major drawbacks and we expect it to be preferred over DPR at most task.

To what degree the new methods will be helpful in actual AR applications however, remains to be tested through further observations and formal user studies. To makes these studies possible and to make different rendering methods for handheld AR more accessible, was the purpose of building this prototype. In the following section we will review various user studies conducted on previous UPR and DPR prototypes and explain which of them would be useful to perform with the new methods. Furthermore, we will cover how to build and perform such studies with our prototype.

3.4 User Studies suited to Evaluate Rendering Methods

After the implementation of the prototype, the next step is to thoroughly test the new rendering methods and to evaluate whether they deliver the expected advantages over UPR and DPR. It is necessary to compare these different rendering methods against each other in formal user studies, which should be designed reflect common interaction and usage patterns in AR applications.

three types of studies to evaluate the rendering methods

Three types of studies focusing on (i) Search and Select, (ii) Cross Context Interaction and (iii) Depth Estimation make up the majority of existing user studies in handheld AR. These types of studies were commonly used to compare UPR to DPR and should cover all important aspects of interaction in AR applications. In the following we will go over representative studies for these three types and develop ways to conduct such studies with our prototype in order to compare the different rendering methods.

in-/dependent variables and questionnaires

Important for all types of studies is which metrics to measure and evaluate. Samini et al. [2017] analyzed a collection of user studies in the field of AR/VR and composed a list of the most commonly used in-/dependent variables. They found the most important independent variables to be: ′interaction technique′, ′display size′ and ‘size of the virtual object′. For the dependent variables: ‘task completion time′, ‘accuracy′ and ‘success rate′. Most studies they analyzed also included questionnaires with five- or seven-point Likert scales. Samini et al. found the most common metrics to be: ‘likeability′ (of each technique), ‘ease of use′, ‘perceived speed′, ‘perceived accuracy′, ‘intuitiveness′ and ‘comfort′. In our case, the primary independent variable would be ‘rendering method′, but it could still be helpful to include ‘display size′ and test each method on both a phone- and tablet-sized lens. Both types of devices are supported by our prototype and testing on both would allow for comparisons with older results from other user studies.

All three study types (Search and Select, Cross Context Interaction and Depth Estimation) can be realized as withinsubjects designs. A between-groups design comparing five different rendering methods and two display sizes would be possible but would require many participants. The order in which the participants use the different rendering methods should be randomized using the Latin Square method.

3.4.1 Search and Select

One of the most common interaction techniques in AR applications is the searching and selecting of virtual objects, because it precedes almost every other action a user may perform on the object. Therefore, a search and select task is commonly chosen to compare different AR devices or techniques [Samini and Palmerius, 2016, Looser et al., 2007, Baričević et al., 2012, Tomioka et al., 2013]. The purpose of this study is to evaluate how difficult it is for participants to use lenses with different rendering methods to search the environment for virtual objects and consequently select them. Similar to the setup in other studies [Samini and Palmerius, 2016, Looser et al., 2007, Baričević et al., 2012, Tomioka et al., 2013], the task should consist of virtual objects appearing in random location in the action space. The action space can either be the whole room or be limited to a tabletop environment (like in the study conducted by Baričević et al. [2012]). Participants must find and select these virtual objects with the lens while different measurements are being taken: (i) the average searching time (the time it takes a user from selecting one object to finding the next one) and (ii) the selection time (the time it takes the user to select an object once it appears in the lens). For some selection techniques it is possible to measure the path deviation during the selection part. We will go over some selection techniques in the following section and discuss which ones are suited best for this kind of study.

Past results of user studies that compared UPR to DPR show that even though UPR is preferred by users for the selection part, the DPR method is significantly faster for searching the environment [Baričević et al., 2012, Samini and Palmerius, 2016]. We hypothesize that the new rendering methods speed up the searching part while keeping the intuitiveness of UPR for the selection part. A questionnaire should be included to collect subjective feedback from participants and evaluate which method is preferred by users.

Selection Techniques

Even though selecting a virtual object by simply touching it on the screen may be the first selection technique that comes to mind, it is not the only one. Examples besides 'on screen touch' include 'direct touch' (touching the virtual object directly with either a hand or a pointing-stick) and 'center select' (the lens selects a virtual object after the user focuses on it hand holds it in the center for a while).

'on-screen touch' is not the only way to select virtual objects

For our search and select study however, it makes sense to focus on only one or maybe two selection techniques, as the main point of the study is to compare the different rendering methods to each other. Samini et al. [2016] compared UPR to DPR in a search and select study using 'on screen touch'. They recognized that the finger occludes large parts of the display when selecting an object. They conducted another study comparing different selection techniques for UPR [2019] and compared 'on screen touch' to the two alternative selection techniques 'center select' and 'icon select' (pressing an icon from a list on the screen, the icon corresponds to the desired virtual object). They discovered another drawback of 'on screen touch', as users reported it to be the most physically tiring. This was likely the case because the device they used was a large tablet and users had to hold the tablet with one hand and select objects with the other. The participants of this study seemed to prefer 'center select' while 'icon select' was the slowest due to the extra time spent to associate the icons with the corresponding virtual objects. These results conform with the results from another user study comparing different selection techniques for head mounted AR [Looser et al., 2007]. In this study participants preferred selecting objects by hovering a handheld lens over them. This technique is very similar to 'center select' and was reported to be the most enjoyable and least physically and mentally demanding. Direct touch was used in a study conducted in a VR simulation

[Baričević et al., 2012], but it is difficult to implement on UPR devices because it would need a real time 3D reconstruction of the user's hand. Therefore, we think that 'center select' is the best technique to conduct a search and select study with. It would also be possible to include 'on screen touch' because it may be easier to use on phone-sized lenses and is probably more likely to be implemented by developers of AR applications.

3.4.2 Cross Context Interaction

The alignment between on screen content and surroundings is not as accurate in the new rendering methods as in true UPR. The purpose of this study is to evaluate how much this affects usability and specifically to what extend it limits the use of surrounding context . This can be tested with a task that requires the user to repetitively cross the border of the magic lens and directly relate the surrounding context to the view inside the lens. Pucihar et al. [2014] compared fixed-POV UPR to DPR in a study focusing on such a task. Participants were asked to move a marker from outside the augmented region into a goal area that was only visible through the lens. An addition to this task could be to have the goal area move within the view of the magic lens. This prevents the user from associating the position of the goal area with a point in the real world and then moving the marker while only focusing on their real-world view. It forces the user to continually switch between the augmented view and the surroundings, until the marker has entered the lens.

A similar task for such a study could consist of moving the magic lens – and the goal area with it – over a real-world marker instead of the other way around. This technique could be more representative for how the lens is used in an actual application. Variables to measure in the experiments are (i) the time it takes the participants to complete the task, (ii) the path deviation once the marker has entered the view of the magic lens and (iii) the success rate. Pucihar et al. observed through their study that the completion time and path deviation were smaller when using fixed-POV UPR over DPR [2014], though this effect was only noticeable when using the larger tablet-sized lens. We hypothesize that the results of AUPR will be the best and that our other rendering methods fall somewhere between AUPR and DPR. Method $M1_{FOV}$ is expected to rank closer to AUPR, while $M2_{Rot}$ will probably be closer to DPR.

3.4.3 Depth Estimation

Depth estimation studies can be helpful to evaluate how easy it is to understand spatial relationships between virtual objects , real-world objects and one-self. Several studies have been conducted to analyze depth perception in AR [Diaz et al., 2017, Dey et al., 2012, Liu et al., 2020], most of them focusing on what cues can make it easier. The typical setup for such a study consists of mounting the device in a fixed position at eye height. The participants are then tasked to estimate the distance of virtual objects by looking at them through the device. The virtual objects can be of different sizes, shapes and color; some may be abstract shapes and other may resemble human beings or other objects. Variables to measure are (i) the error and (ii) the time it takes to make an estimation.

Our prototype allows for simple drop shadows of virtual objects, which were found to be one of the primary depth cues along with simply the size of the object [Diaz et al., 2017, Dey et al., 2012]. This should make it possible to estimate distances to virtual objects more accurately. Depth cues, however, are only of secondary importance for this study. The focus lies on comparing the different rendering methods to each other; the available depth cues will be the same for all methods. There are no existing studies focusing on depth perception in UPR versus DPR, which makes it hard to predict how the new rendering methods will perform in comparison to DPR. We do however expect that distances will be slightly underestimated, as that seems to be a common result among similar studies [Diaz et al., 2017, Dey et al., 2012, Liu et al., 2020].

3.4.4 Conducting Studies with the Prototype

Our prototype includes a demo scene and corresponding script, which showcases different functionalities that will be needed to perform these user studies. It enables easy switching between the rendering methods and manipulation of different variables like the FOV. It shows how to spawn virtual objects by the click of a button, how to select those virtual objects and how to display their distances to the user. Two code excerpts are given below:

Spawning a virtual cube at the position of the device:

```
GameObject newCube = Instantiate<GameObject>(
                       cubePrefab,
                       arCamera.transform.position,
                       Quaternion.identity);
```

Selecting a virtual object and calculating its distance to the user:

```
Ray ray = eyesCamera.ScreenPointToRay(Input.mousePosition);
RaycastHit hitData;
if (Physics.Raycast(ray, out hitData, 1000)){
      GameObject objHit = hitData.transform.gameObject;
      if (objHit != null && objHit.tag == "selectable"){
            float distance = (objHit.transform.position -
                              trackedPositionEyes).magnitude;
      }
}
```

Functionalities like these act as sample implementations and as a base on which further functionality can be built. The code base is structured similarly to the explanation of our implementation in the previous chapter, which makes it easy to go back and forth between the two to fully understand them. All this makes it easy to build and execute the user studies we presented in this chapter and makes it possible to use the new rendering methods for future studies or applications in the field of handheld AR.

Chapter 4

Summary and Future Work

In this last chapter, the ideas, tasks and results of this thesis are summarized and a brief outlook on future steps and potential improvements to the prototype is given.

4.1 Summary and Contributions

short summary

This thesis deals with the introduction and implementation of new rendering methods for handheld Augmented Reality (AR). After analyzing existing rendering methods, three new methods, combining aspects form user- and device-perspective rendering, are proposed. These five methods are then implemented in the form of an application for iOS. Finally, past user studies in the field of handheld AR are analyzed and condensed into three study designs, which intend to compare the different rendering methods and can be conducted using our prototype.

problem explanation

Handheld devices are one of the most common ways in which AR is used today. In handheld AR, the combination of real-world view and virtual information is achieved by superimposing the virtual scene onto an image of the real world. This image is captured by a camera on the back of the device, which is why this process is called device-perspective rendering (DPR). It results in a misalignment between what is displayed on screen and the view of the user because the displayed image is completely dependent on the position and orientation of the device. In user-perspective rendering (UPR), on the other hand, the position of the user's eyes is taken into account and the displayed image is rendered from the perspective of the user. UPR comes with benefits and drawbacks, which creates a trade-off between itself and DPR.

contributions

In this thesis, three new rendering methods are introduced. They aim to combine the advantages of DPR and UPR by combining aspects from both sides. One of the new methods ($M1_{FOV}$) works much like UPR but with an increased field of view (FOV). Method two ($M2_{Rot}$) displays almost the entire camera image like DPR but responds to movements in the user's head position by rotating. The third method ($M3_{Int}$) interpolates the point of view between the position of the user and the device's camera. These new rendering methods, including approximated UPR and DPR, were implemented in the form of an application for iOS and a detailed explanation of their implementation is given in this thesis. The prototype was built in Unity and is intended to enable direct comparisons between different rendering methods. To the best of our knowledge it is the first to work without requiring any additional hardware and runs on of-the-shelf consumer smartphones and tablets. This makes our implementation of these rendering methods accessible for future applications and research in the field of handheld AR. First observations, through informal testing with a small group of users, make the new rendering methods seem promising. Nonetheless, they will need to be tested in formal user studies. In order to design user studies that are suited to thoroughly test and compare the different rendering methods, previous user studies in the field of AR were reviewed and combined into the designs of three types of user studies, which can be conducted using our prototype.

4.2 Future Work

conduct user studies

Naturally, the next step is to conduct the user studies that are presented at the end of this thesis. For this, some small customizations can be added to the prototype, such as other selection techniques or custom models for virtual objects.

explore variations of new methods

Depending on the results of these studies, it would be interesting to explore more variations of the different rendering methods. Instead of choosing a constant FOV, it would be possible to dynamically change the size of the displayed and augmented area. This way, the FOV could be automatically adjusted to fit a virtual object on the screen, without being too wide. Alternatively, the FOV could be scaled relative to the speed with which the user moves the device. A fast movement speed suggests that the user is searching for virtual objects in the scene, which a larger FOV will be helpful for. However, when the user is focusing on a single point in space, a smaller FOV will deliver the best alignments.

future improvements to the prototype

Additionally, new improvements to the prototype will be possible through updated versions of Apple's AR Kit and Unity's AR Foundation. With Unity's support for AR Kit 6, features like access to the 4K camera image and occlusion of virtual objects by humans will be possible. In future versions it may also be possible to simultaneously access the front and wide-angle rear cameras, which will allow for larger viewing angles from the side of the device.

Appendix A

Prototype Project Files

Our implementation (in the form of its Unity Project Folder) is provided alongside this work on a USB stick.

It is also available on GitHub [1].

[1]github.com/JoWilhelm/UPRLens

Bibliography

Ronald T. Azuma. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 08 1997. doi: 10.1162/pres.1997.6.4.355. URL `https://doi.org/10.1162/pres.1997.6.4.355`.

Domagoj Baričević, Cha Lee, Matthew Turk, Tobias Höllerer, and Doug Bowman. A hand-held ar magic lens with user-perspective rendering. pages 197–206, 11 2012. ISBN 978-1-4673-4660-3. doi: 10.1109/ISMAR.2012.6402557.

Domagoj Baričević, Tobias Höllerer, Pradeep Sen, and Matthew A. Turk. User-perspective augmented reality magic lens from gradients. In Rynson W. H. Lau, Dinesh Manocha, Taku Komura, Aditi Majumder, and Weiwei Xu, editors, *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology, VRST 2014, Edinburgh, United Kingdom, November 11-13, 2014*, pages 87–96. ACM, 2014. doi: 10.1145/2671015.2671027. URL `https://doi.org/10.1145/2671015.2671027`.

Domagoj Baričević, Tobias Höllerer, Pradeep Sen, and Matthew Turk. User-perspective ar magic lens from gradient-based ibr and semi-dense stereo. *IEEE Transactions on Visualization and Computer Graphics*, 23:1–1, 04 2016. doi: 10.1109/TVCG.2016.2559483.

Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: The see-through interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, page 73–80, New York, NY, USA, 1993. Association for Computing Machinery. ISBN 0897916018.

doi: 10.1145/166117.166126. URL `https://doi.org/10.1145/166117.166126`.

A. Dey, M. Billinghurst, R.W. Lindeman, and J.E. II Swan. A systematic review of 10 years of augmented reality usability studies: 2005 to 2014. 2018. doi: 10.3389/frobt.2018.00037.

Arindam Dey, Graeme Jarvis, Christian Sandor, and Gerhard Reitmayr. Tablet versus phone: Depth perception in handheld augmented reality. 11 2012. doi: 10.1109/ISMAR.2012.6402556.

Catherine Diaz, Michael Walker, Danielle Albers Szafir, and Daniel Szafir. Designing for depth perceptions in augmented reality. pages 111–122, 10 2017. doi: 10.1109/ISMAR.2017.28.

Alex Hill, Jacob Schiefer, Jeff Wilson, Brian Davidson, Maribeth Gandy, and Blair Macintyre. Virtual transparency: Introducing parallax view into video see-through ar. pages 239–240, 10 2011. doi: 10.1109/ISMAR.2011.6092395.

M. Kanbara, T. Okuma, H. Takemura, and N. Yokoya. A stereoscopic video see-through augmented reality system based on real-time vision-based registration. In *Proceedings IEEE Virtual Reality 2000 (Cat. No.00CB37048)*, pages 255–262, 2000. doi: 10.1109/VR.2000.840506.

H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pages 85–94, 1999. doi: 10.1109/IWAR.1999.803809.

Ernst Kruijff, J.E. II Swan, and Steven Feiner. Perceptual issues in augmented reality revisited. In *2010 IEEE International Symposium on Mixed and Augmented Reality*, pages 3–12, 2010. doi: 10.1109/ISMAR.2010.5643530.

Victor Kyriazakos and Konstantinos Moustakas. A user-perspective view for mobile ar systems using discrete depth segmentation. In *2015 International Conference on Cyberworlds (CW)*, pages 69–72, 2015. doi: 10.1109/CW.2015.67.

May Liu, Gayathri Narasimham, Jeanine Stefanucci, Sarah Creem-Regehr, and Bobby Bodenheimer. Distance perception in modern mobile augmented reality. pages 196–200, 03 2020. doi: 10.1109/VRW50115.2020.00042.

Julian Looser, Mark Billinghurst, and Andy Cockburn. Through the looking glass: The use of lenses as an interface tool for augmented reality. 01 2004. doi: 10.1145/988834.988870.

Julian Looser, Mark Billinghurst, Raphael Grasset, and Andy Cockburn. An evaluation of virtual lenses for object selection in augmented reality. pages 203–210, 01 2007. doi: 10.1145/1321261.1321297.

M. Mohring, C. Lessig, and O. Bimber. Video see-through ar on consumer cell-phones. pages 252– 253, 12 2004. ISBN 0-7695-2191-6. doi: 10.1109/ISMAR.2004.63.

Klen Pucihar, Paul Coulton, and Jason Alexander. Evaluating dual-view perceptual issues in handheld augmented reality: Device vs. user perspective rendering. pages 381–388, 12 2013. doi: 10.1145/2522848.2522885.

Klen Pucihar, Paul Coulton, and Jason Alexander. The use of surrounding visual context in handheld ar. 04 2014. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557125.

Jun Rekimoto and Katashi Nagao. The world through the computer: Computer augmented interaction with real world environments. *UIST (User Interface Software and Technology): Proceedings of the ACM Symposium*, 11 1995. doi: 10.1145/215585.215639.

Ali Samini and Karljohan Lundin Palmerius. Wand-like interaction with a hand-held tablet device—a study on selection and pose manipulation techniques. *Information*, 10(4), 2019. ISSN 2078-2489. doi: 10.3390/info10040152. URL `https://www.mdpi.com/2078-2489/10/4/152`.

Ali Samini and Karljohan Palmerius. A user study on touch interaction for user-perspective rendering in hand-held video see-through augmented reality. pages 304–317, 06 2016. ISBN 978-3-319-40650-3. doi: 10.1007/978-3-319-40651-0_25.

Ali Samini and Karljohan Lundin Palmerius. A perspective geometry approach to user-perspective rendering in hand-held video see-through augmented reality. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, VRST '14, page 207–208, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450332538. doi: 10.1145/2671015.2671127. URL `https://doi.org/10.1145/2671015.2671127`.

Ali Samini and Karljohan Lundin Palmerius. Popular performance metrics for evaluation of interaction in virtual and augmented reality. In *2017 International Conference on Cyberworlds (CW)*, pages 206–209, 2017. doi: 10.1109/CW.2017.25.

Ivan E. Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), page 757–764, New York, NY, USA, 1968. Association for Computing Machinery. ISBN 9781450378994. doi: 10.1145/1476589.1476686. URL `https://doi.org/10.1145/1476589.1476686`.

A. Takagi, S. Yamazaki, Y. Saito, and N. Taniguchi. Development of a stereo video see-through hmd for ar systems. In *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, pages 68–77, 2000. doi: 10.1109/ISAR.2000.880925.

Makoto Tomioka, Sei Ikeda, and Kosuke Sato. Approximated user-perspective rendering in tablet-based augmented reality. pages 21–28, 10 2013. doi: 10.1109/ISMAR.2013.6671760.

John Viega, Matthew J. Conway, George Williams, and Randy Pausch. 3d magic lenses. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, UIST '96, page 51–58, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917987. doi: 10.1145/237091.237098. URL `https://doi.org/10.1145/237091.237098`.

Daniel Wagner and Dieter Schmalstieg. First steps towards handheld augmented reality. pages 127– 135, 11 2003. ISBN 0-7695-2034-0. doi: 10.1109/ISWC.2003.1241402.

Matthias Wloka and Eliot Greenfield. The virtual tricorder: A uniform interface for virtual reality. In *Proc. of UIST '95*, pages 39–40, 1995.

Index

Typeset September 9, 2022