Università di Genova

DIPARTIMENTO DI INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

A Registration Framework for the Comparison of Video and Optical See-Through Devices in Interactive Augmented Reality

Giorgio Ballestin

Theses Series DIBRIS-TH-2021-42

Università di **Genova**

Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi

Ph.D. Thesis in
Computer Science and Systems Engineering
Computer Science Curriculum

A Registration Framework for the Comparison of Video and Optical See-Through Devices in Interactive Augmented Reality

by

Giorgio Ballestin

May, 2021

Ph.D. Thesis in Computer Science and Systems Engineering (S.S.D. INF/01)
Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi
Università di Genova

Candidate
Giorgio Ballestin
`giorgio.ballestin@dibris.unige.it`

Title
A Registration Framework for the Comparison of Video and Optical See-Through Devices in Interactive Augmented Reality

Advisors
Fabio Solari
DIBRIS, Università di Genova
`fabio.solari@unige.it`

Manuela Chessa
DIBRIS, Università di Genova
`manuela.chessa@unige.it`

External Reviewers
Carlos F. Crispim-Junior
Université Lumière Lyon 2
`carlos.crispimjunior@univ-lyon2.fr`

Lorenzo Natale
Istituto Italiano di Tecnologia
`lorenzo.natale@iit.it`

Location
DIBRIS, Univ. di Genova
Via Dodecaneso, 35
I-16146 Genova, Italy

Submitted On
May 2021

Abstract

Augmented Reality (AR) is a technology that has been growing in interest in the past decade. Many factors are currently hindering its distribution to the general public. The heavy processing requirements, hardware limitations, and the need for an easy to use portable/wearable device are still problems to be addressed. The AR field is currently split between the use of widespread devices (smartphones) for easily deployable applications, and the use of high-end Head Mounted Displays (HMD), which are generally very expensive, and often require a cumbersome tethered high-end computer rig to the side. In the next evolution of Wearable devices and Internet of Things, Augmented Reality can play a key role in the human-computer interaction field. The potential of having access to computational capabilities embedded in a simple pair of glasses is huge, from professional applications (in medicine, industry, design) to every-day's life. Augmented Reality problematics are strictly related to the way the registration and visualization are performed. During registration, a model of the environment is obtained through several sensors (often with cameras). The obtained model is then used to digitally augment the image with additional images before conveying the merged view to the user. This process, however, is still far from being perfect. The environmental registration is currently a computationally complex task, which leads to very simplified models, that often hinder the development space. Different visualization techniques also have a different impact on the users, due to the introduction of parallaxes and latencies, which cause several artifacts and perception issues.

To this aim, we developed a registration framework that can be used to develop augmented reality environments, having all the real (including the users) and virtual elements co-localized and registered in a common reference frame. Specifically, several devices are calibrated and aligned in a common reference frame, and measurements are captured with an external independent common measurement system. The proposed framework is based on methodologies that can be used for any device.

We then used the proposed framework to create AR scenarios, to assess the optical and perceptual differences of different types of AR HMDs and their impact on the interactivity. The methodology involves the design of several experimental sessions under rigorous, repeatable conditions, and the subsequent evaluation of performances. Residual errors, user experiences end performances were evaluated considering both quantitative and qualitative metrics derived from the collection and the analysis of heterogeneous, unbiased data, and self-assessment questionnaires.

Our results show that depth perception appears to be compressed when using several AR HMDs, potentially hindering the interaction with AR environments. The effect is particularly prominent when fewer cues and feedbacks are provided. If the users are able to perform a visual alignment between the real and virtual geometries, however, an effective interaction can be achieved, even if the overlap is not perfect.

Publications

Some ideas and figures have appeared previously in the following publications:

JOURNAL PUBLICATIONS

Giorgio Ballestin, Manuela Chessa, and Fabio Solari. 'A Registration Framework for the Comparison of Video and Optical See-Through Devices in Interactive Augmented Reality'. In: IEEE Access, 2021.

CONFERENCE PROCEEDINGS

Giorgio Ballestin, Manuela Chessa, and Fabio Solari. 'Assessment of Optical See-Through Head Mounted Display Calibration for Interactive Augmented Reality'. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019.

Giorgio Ballestin, Fabio Solari, and Manuela Chessa. 'Perception and action in peripersonal space: A comparison between video and optical see-through augmented reality devices.' In: IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), 2018.

Giorgio Ballestin, Chiara Bassano, Fabio Solari, and Manuela chessa. 'A Virtual Reality Game Design for Collaborative Team-Building: A Proof of Concept.' In: Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization, 2020.

Ivan Valentini, Giorgio Ballestin, Chiara Bassano, Fabio Solari and Manuela Chessa. 'Improving Obstacle Awareness to Enhance Interaction in Virtual Reality'. In: IEEE Conference on Virtual Reality and 3D User Interfaces (VR), 2020.

POSTERS AND ORAL PRESENTATIONS

Chiara Bassano, Giorgio Ballestin, Eleonora Ceccaldi, Fanny Isabelle Larradet, Maurizio Mancini, Erica Volta and Radoslaw Niewiadomski. 'A VR Game-Based System for Multimodal Emotion Data Collection'. In: Motion, Interaction and Games. MIG '19. Newcastle upon Tyne, United Kingdom: Association for Computing Machinery, 2019.

Acknowledgements

First of all, I would like to thank my supervisors Manuela Chessa and Fabio Solari, for their support, their time and for sharing their experience and expertise with me.
Then, many thanks to everybody who kindly gave me their time, patience and energy to participate in my experiments, receiving no reward!

Contents

List of Figures

List of Tables

Acronyms

AR - Augmented Reality

CAD - Computer Aided Design

DOF - Degree of Freedom

EEG - Electroencephalography

EMG - Electromyography

FOV - Field of View

HCI - Human Computer Interaction

HMD - Head-mounted Display

HPM - Human Processor Model

HSV - Hue Saturation Value

IPD - Interpupillary Distance

IPQ - IGroup Presence Questionnaire

IR - Infrared

LCD - Liquid Crystal Display

MAE - Mean Average Error

MHP - Human Processor Model

MOCAP - Motion Capture

MR - Mixed Reality

MRE - Mean Reprojection Error

NHCI - Natural Human Computer Interface

OST - Optical See-Through

PCA - Principal Component Analysis

QHD - Quad High Definition

RAM - Random Access Memory

RANSAC - Random Sample Consensus

RGB - Red Green Blue

RGB-D - Red Green Blue-Depth

RW - Real World

SD - Standard Deviation

SDK - Software Development Kit

SLAM - Simultaneous Localization And Mapping

SPAAM - Single Point Active Alignment Method

SSQ - Simulator Sickness Questionnaire

SVD - Singular Value Decomposition

UDP - User Data Protocol

UI - User Interface

UX - User Experience

VR - Virtual Reality

VST - Video See-Through

WIMP - Windows Icons Menus Pointer

1 Introduction

Augmented Reality is a multidisciplinary research area which resides in the human-computer interaction field. To understand the mechanisms behind human memory, perception and the sensory system, it may be necessary to delve into several fields, like psychology, neurobiology, and other cognitive sciences. Some issues are related to ergonomics (both cognitive and physical). Finally, the hardware and implementation require competences in several computer science sub-fields, like computer graphics, computer vision, artificial intelligence. Complementary requirements are also knowledge in graphic design and aesthetics, in order to properly design graphical interfaces that can properly deal with the information/knowledge visualization.

Augmented Reality has had a widespread success in the last decade, however, being a constantly evolving sector, research in this area is very rich and diversified, with lots of open questions, issues to be solved and aspects requiring further investigations, from hardware implementation to graphics, from user experience to interaction. Furthermore, with the diffusion of this new technology, the areas of application are multiple and diverse. The following part, thus, aims at giving the reader a complete overview of the specific context and motivation of the research work presented in this thesis and highlights its contributions.

1.1 Context

One of the key aspects of Augmented Reality is achieving a seamless integration of virtual and real sensory information. This has proven to be all but a trivial task for pretty much every sensory modality; when considering vision only, the current commercial solutions are still facing several problematics. A seamless visual alignment of the AR content requires a perfect knowledge of the optical model, which includes the user's vision system (which, of course, changes for every user) and the distortions introduced by the visualization device itself. The surrounding environment must also be tracked or reconstructed, which requires advanced SLAM capabilities, and a non-trivial computational cost. Indeed, to be able to run computationally heavy algorithms, some devices must be tethered to an external PC, hindering the interactivity and somewhat going against their wearable nature.

The spread of AR research has led to many different kinds of visualization devices, such as handheld ones (smartphones, tablets), HMDs, smart glasses, and so on. Researchers thus started to investigate the differences between different engineering approaches and solutions to convey the visual augmentation. Specifically, different visualization devices, having a different optical model, potentially lead to a different perception of the real-virtual overlap, which thus also leads to interaction differences within the AR environment.

Experimenting with several different AR devices, however, is not trivial, as different HMDs usually do not share the same tracking systems, and often lack a robust spatial tracking system for the user interaction. It is therefore necessary that all the devices are calibrated and aligned (registered) in a common reference frame, and that measurements are done with an external independent common measurement system. The current work focuses on the perceptual differences between optical see-through (OST) and video see-through (VST) head-mounted displays (HMD), and the implications of those differences in the perception-interaction loop. The context of this work is thus in a research setting, but results found can be generalized to any context where multiple AR devices are required, e.g. in shared (multi-user) AR.

1.2 Motivation

Knowledge workers, technicians, manufacturing employees, are required to perform data-intensive tasks every day. Accessing, analyzing, and acting on large quantities of information on several different devices. For employees in a number of industries, AR has the potential to be a game-changer in how they can interact with and share information, in a more natural way.

For instance, employees on a factory floor would be able to view instructions in real-time—when and where it's contextually appropriate—instead of flipping through a cumbersome and distracting training manual. Utilizing AR technology will enable employees to reduce time, cost, and error. According to the Augmented Reality for Enterprise Alliance [3], "As enterprises begin to rapidly develop connected infrastructure, from manufacturing to logistics, and ultimately to the consumer, massive amounts of data are being collected and used for analysis. AR can provide a data to human interface, allowing workers, managers and executives to see the world augmented with a rich dataset."

Forrester Research [30] reports that nearly 14 million workers will utilize a wearable by 2025, up from 400,000 workers in 2017, representing a 57% growth per year. A few companies have already started to explore AR's potential as a technological investment that will reward in terms of efficiency, productivity, training, and a number of other relevant use cases.

For those brands who want to deliver richer customer experiences and increase sales, AR offers plenty of opportunities to transform the way that customers purchase products. For instance, IKEA leveraged AR's technology to enable customers to test drive their products in real-time by providing them use of an AR catalogue app that allowed them to visualize how a piece of furniture would look in their home prior to purchase, utilizing the technology's immersive capabilities to create a differentiated purchase experience that's tailored to customer service needs. Combining improved customer service with a technology that removes barriers from the purchase pathway directly leads to increased sales.

Another useful ability of AR's technological potential for the enterprise revolves around its 3D technology. Designers and creatives could interact with highly immersive 3D visualizations that will allow them to refine models in real-time, ultimately communicating what a finished product is like in a more realistic manner than the flatness of 2D designs. Businesses in a number of different industries could benefit from this enhanced model visualization: enabling designers, architects, city planners and others to validate design ideas early and often, ultimately saving these individuals and their companies' time and effort.

With this research, I am planning to explore some of the core problematics tied to this technology. Some of them are caused by the way humans perceive visual information, and how to adjust the provided stimulus to best fit our visual system. Then, we have the interaction aspect, thus how to achieve an affordable interaction taking into account the bio-mechanical restrictions of the human body and the technological limitations of the devices (e.g. lack of haptic feedback). Moreover, we must investigate the possible benefits of using this technology as aB tool for both professional use and general public applications (e.g. to increase productivity, learning, and so on). The functionalities of AR devices are also driven by their ability to gather all the required information through visual data, thus needing vast use of computer vision techniques as well.

1.3 Contribution

The main contribution of this thesis is a registration framework that can be used to develop augmented reality environments, having all the real (including the users) and virtual elements co-localized and registered in a common reference frame. The framework is divided in several modules which handle the registration of all the devices and tracking systems in the same reference frame, in such a way that all the data collected can be coherently compared. All the software modules are provided in a GitHub repository, including the instructions to setup an experimental scene in Unity, and all the 3D printable files and electronic schematics used in this study. The code documentation, with the explanation of each class parameter and usage instructions, is provided as appendix to this thesis.

The framework has been validated with quantitative parameters derived from the analysis of the accuracy and residual error of the different elements of the setup, i.e. for the OST HMD we considered the residual reprojection errors of the stereo calibration between the virtual cameras representing the user's eyes and the real cameras of a mannequin that wore the HMD to collect the data, the misalignment error (expressed as Euclidean distance) between the points a virtual checkerboard and a real (tracked) one, and its distribution across the plane orthogonal to the optical axis of the user. For the user's interaction tracking device, we measured the accuracy and precision of the tracked finger tip 3D world positions with respect to the baseline obtained with the (more precise) HTC Vive Lighthouse system. The proposed methodologies and modules are independent to each other and can be combined and generalized to work with different HMD or tracking sensors (e.g. MOCAP).

We then developed several experimental sessions to investigate the performance and user experience of users of different AR devices. The main research question was to establish the perceptual differences with the focus on spatial perception (e.g. depth perception) when using different HMDs, and to establish what cues allow the users to achieve an effective interaction. We considered three cases: in the first one was based on reaching tasks, to observe the differences between a monocular VST and a stereo OST HMDs. In this case, the devices were not synchronized with each other, hindering the obtained data analysis. The second case was a blind reaching task which has been performed with the proposed framework in a controlled setting. The aim of this experiment was to isolate the visual perceptual error in the reaching task by removing the visual alignment cue by using the blind paradigm. The last case, also carried out with the proposed framework, was an interaction task, where people had to move objects in a scene containing both real and virtual elements.

The data obtained during these experimental sessions is composed both of quantitative and qualitative data such as the positions and rotation of the user's hands, head, and fingertips, and self-assessment validated questionnaires. We thus evaluated both the accuracy and precision of the user's interaction, and their user experience, in terms of immersion and development of cyber-sickness and eye fatigue symptoms.

The work presented in this thesis has been the subject of several peer-reviewed publications. Specifically, the whole study has been published on IEEE Access [8], with the title *'A Registration Framework for the Comparison of Video and Optical See-Through Devices in Interactive Augmented Reality'* . The OST HMD registration, described in Section 3.0.1, has also been published in a separate study *'Assessment of Optical See-Through Head Mounted Display Calibration for Interactive Augmented Reality'* at the proceedings of the IEEE International Conference on Computer Vision Workshops (2019) [7]. Finally, the experimental setup described in Chapter 4.1 has been published at IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct) (2018), with the title *'Perception and action in Peripersonal Space: A Comparison Between Video and Optical See-Through Augmented Reality Devices'* [9].

1.4 Outline

In the next chapters, first, I will briefly introduce state of the art technologies and software frameworks currently available for AR, devices and approaches for interaction, highlighting their pros and cons, potentialities and limitations (Chapter 2).

Then, I will explain how my research on the development of an AR registration framework (Chapter 3) has been articulated, with particular attention to the specifically developed methodologies and software modules, and I will present obtained results.

Chapter 4 will then describe three studies, performed for the evaluation of human perception and natural human computer interaction, which have been designed using the proposed framework described in chapter 3.

Finally, I will discuss the obtained findings with respect to the original research questions (Chapter 5) and will identify some open issues and possible future developments (Chapter 6).

The devised registration framework is publicly available online [6] licensed under *The Unlicense* (a license with no conditions whatsoever which dedicates works to the public domain [1]): unlicensed works, modifications, and larger works may be distributed under different terms and without source code. The repository documentation is provided in the appendix section of this document.

2 Background

The following part aims to include all the state of the art necessary to the understanding of the research work described in this thesis. As different topics were involved, i.e. perception and interaction within an augmented environment and User Experience Assessment, this part incorporates multidisciplinary and heterogeneous information.

2.1 AR Devices

AR systems started to spread in commercial applications in the past decade, but the first working prototypes have emerged since the 90's. AR systems create an experience where the real world is enhanced by computer-generated information, usually with a combination of different sensory modalities such as visual, auditory and haptic. AR is often confused with Virtual Reality (VR): the difference is that in a VR experience, the user is immersed in a completely virtual world, while in an AR experience the computer-generated information is blended with the physical world. Both technologies reside on the Reality-Virtuality continuum first introduced by Paul Milgram [60] to disambiguate between different compositions of virtual and real information. Sometimes AR and Mixed Reality (MR), which includes the whole spectrum from Augmented Reality to Augmented Virtuality, are used as synonyms, to simply disambiguate from completely real and completely virtual (VR) experiences.

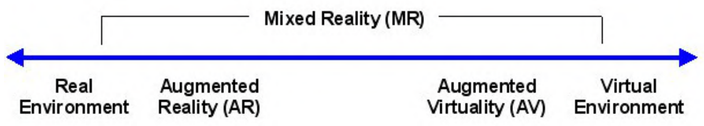


Figure 1: In the Reality-Virtuality continuum introduced by Paul Milgram, AR resides in the Mixed Reality area, where the majority of the information comes from the real physical environment.

AR has also been defined [104] as a system that fulfills three basic features: a combination of real and virtual worlds, real-time interaction, and accurate 3D registration of virtual and real objects. By that definition, many devices which are commonly referred as AR (e.g. smart glasses such as Google Glass) cannot be truly considered AR devices, since they lack any form of environment registration: the displayed augmentation is completely incoherent with the real world, thus the benefits over using a normal display are merely the ability to keep the hands free for other operations (e.g. during surgeries) and the possibility of keeping focus on the current task (e.g. while driving).

Every AR system that aims at believably blending real and virtual information must thus be composed by two main modules: the first one addresses the environment registration, and thus creates a model of the real world using measurements from some kind of sensor (e.g. RGB/RGB-D cameras, inertial, GPS). The second one uses the model obtained during the registration phase to blend (in the most realistic way possible) digital information to the scene, aligning the digital information to the model and maintaining the alignment over time. Real-time interaction can then be performed by leveraging on these two systems. The model obtained during the environmental registration can have different forms and complexities, and is not necessarily computed in real-time. In some cases the model can be as simple as a single plane aligned to the floor plane or to a tracked QR code, which can both be detected by with a camera. To build more complex AR applications, the full 3D reconstruction of the environment (possibly enriched by contextual information) may be required instead.

The most common sensory modality virtually enhanced in AR is the visual one, thus the most widespread AR devices for single users are HMDs, handheld (e.g. smartphones or tablets), or Heads-up displays (HUDs). Regarding AR applications where multiple users are involved, current approaches involve using either several devices registered in the same common reference frame, or a single system (e.g. a projector) to display the visual information to everybody. The latter case is usually referred as spatial AR (SAR).

The different devices available for AR differ for several factors, due to the different way the environment is registered and the way the digital information is blended and conveyed to the user. AR HMDs have been hugely researched in the past years and still are, at the moment, possibly the only option to produce AR applications where the user performs complex interactions with the augmented environment. Depending on the optical system (see Figure 2), AR HMDs are usually referred either as Optical See-Through (VST) or Video See-Through (VST).

In OST HMDs, the augmentation is usually projected on semi-transparent mirror lenses, and the user can see the real world directly through the lenses. Notable examples are the Microsoft Hololens 2, the Magic Leap One, the Meta2, or the Epson Moverio BT-300.

In VST HMDs, the video feed of one or more front facing cameras is portrayed in a display in front of the user's eyes, together with the digital augmentation. Most VR HMDs can thus be converted to AR VST HMDs by adding at least one camera to the HMD. Handheld devices, such as smartphones or tablets, also works as VST.

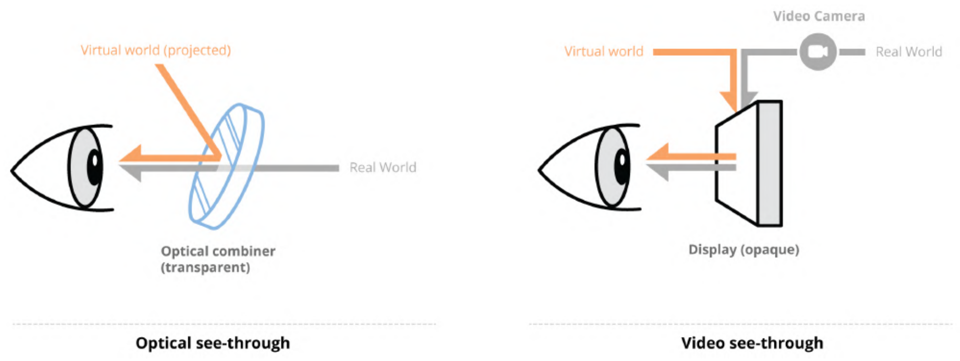


Figure 2: The two main techniques used by AR HMDs to blend virtual and real images: through semi-transparent mirrors (OST) or by displaying the video feed of a camera to a screen in front of the user eyes (VST).

Since this study is focused specifically on AR HMDs, in the next sections the differences between VST and OST HMDs will be discussed in detail.

2.1.1 *Optical See-Through Head Mounted Display (OST HMD)*

Most manufacturer at the moment are leaning towards OST HMD designs, such as the Microsoft Hololens, the Meta2 by Metavision, the Magic Leap or the Epson Moverio BT-200 (See Figure 3).

An advantage of OST HMD is a lower restriction of the user's field of view (FoV), which doesn't necessarily mean the augmentable area is bigger than VST HMDs: rather, the user can still see through the unaugmentable areas of the transparent lenses.

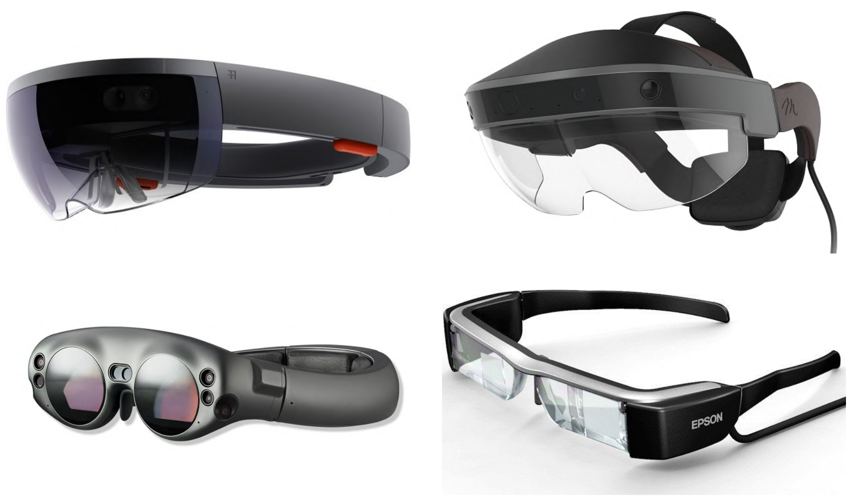


Figure 3: Some examples of commercially available OST HMDs: Microsoft Hololens (top left), Meta2 by Metavision (top right), Magic Leap (bottom left) and Epson Moverio BT-200 (bottom right).

OST HMDs are also safer to use regarding critical-case scenarios, as a power or system failure would not completely blind the user. However, the semi-reflective nature of the lenses blocks some of the light, producing similar viewing conditions of sunglasses or tinted shade glasses, which may be undesirable. Different OST HMDs reflect a varying amount of light (e.g. notice the difference between the lenses shade between the Hololens and Meta2 in 3). More reflective lenses are able to better occlude the real world with the projected imagery, i.e. holograms are perceived as being less translucent, but blocks more light from the real environment, and vice versa.

With OST HMDs, the impact of latency is different than VST HMDs: while in VST the refresh rate of the camera feed is fixed (synchronized) for both the real and virtual content, on OST HMD the latency obviously affects only the virtual projection. This may break the spatial blend perceived, until the user's head slows down and allow the HMD to catch up, which somewhat causes an interruption of the achieved suspension of disbelief.

Some OST HMDs needs to be tethered to an external computer due to several reasons such as computational requirements or heat dissipation (e.g. Meta2), but the HMDs tracking is usually performed through SLAM (Simultaneous Localization and Mapping) algorithms performed with the HMD sensors alone. Several HMDs (e.g. Hololens, Magic Leap) already evolved to the fully wearable state, which is a key usability condition for every large scale AR applications such as in industrial contexts.

An advantage of OST HMDs is that it leads to a lower amount of parallaxes in the optical system with respect to the VST HMD: indeed, in VST HMDs the projection of the real world is captured by a camera which does not have the same optical center neither of the user's eyes nor of the displays through which the camera feed is displayed. In OST HMDs, the real world projection is merely passing through a semi-transparent lens with minimal distortion, which by itself should lead to a lower impact on the user strain. Assuming the lens distortion is minimal or comparable to normally worn glasses, only one parallax can be introduced, if the transformation between the HMD tracking (e.g. the camera optical center, if it is being tracked visually) and the user's eye is unknown (i.e. the HMD is not calibrated). More details on the calibration of OST HMDs will be discussed later, in section 2.4.

2.1.2 *Video See-Through Head Mounted Display (VST HMD)*

AR VST HMDs are often considered a natural successor to VR HMDs: in many cases, such as the ZED mini or Google Cardboard (Figure 4), the enabling technology is just a VR HMD with one or more attached frontal cameras. This type of configuration is easily deployable as VR HMDs are much more widespread than AR ones, which usually are more expensive. For example, in a multi-user scenario, several cheap VR HMDs can be easily converted into VST AR HMDs with minimal cost. The trend for specialized AR HMDs manufacturer however seems to be leading towards the OST approach, especially wearable all-in-one solution (e.g. Hololens). Whereas VR can be used to train professionals to handle dangerous situations in a safe environment, AR usually finds its context of application not in simulation but as an assistive tool. Using a VST HMD in a potentially hazardous scenario poses the additional threat of completely depriving the user of the visual information, in case of malfunctions of the HMD. Many VR HMDs, moreover, require an external tracking system or are tethered to an external PC (e.g. HTC vive, Oculus Rift), completely restricting the usability in open spaces (e.g. walking inside an industrial complex). Recent VR HMDs, however, also started to opt for untethered all-in-one solutions (e.g. Oculus Quest) which may lead to more manufacturers to either make specialized AR-only untethered standalone VST HMDs or directly designing VR ones to be AR-enabled.

Perceptually, VST HMDs are prone to several problematics. First of all, similarly to VR HMDs, they are subject to the vergence-accommodation conflict. This is caused by the mismatch of two depth cues, the disparity of the two views displayed inside the HMD (which causes the eyes vergence movement), and the focusing distance (which triggers the eye's accommodation mechanism), which is fixed to the eye-lenses distance. This can lead to eyestrain, visual fatigue, and focusing problems. Some studies [41, 43] proposed techniques to circumvent this conflict, i.e. by using near-eye light field displays, but all of

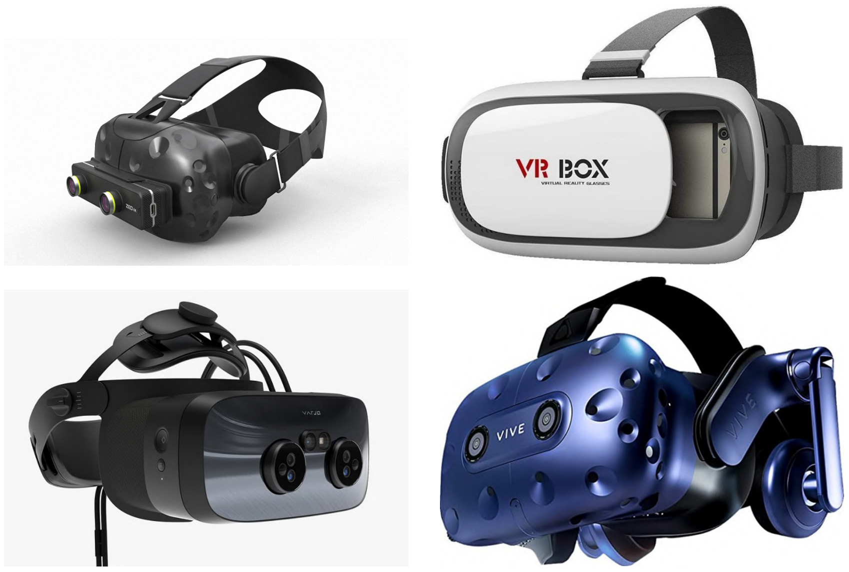


Figure 4: Some examples of commercially available VST HMDs: ZED mini with HTC Vive/Oculus Rift (top left), Google Cardboard (top right), Varjo XR-3 (bottom left), Vive Pro (bottom right).

them are unsuitable for commercial HMDs, as the limitations imposed on the hardware (such as a lower resolution) would severely limit its usability.

Geometrical aberrations in VST HMDs arise from the parallaxes between the several optical elements of the user's eyes-HMD-world model: the projection of the world is distorted due to the passage through the frontal acquisition cameras and then through the visualization display before reaching the user's eye [15, 19]. Some differences, such as the distortions introduced by the lenses or mismatches between the intrinsic parameters, can be solved through calibration or by software compensation [56, 57, 72]. The mismatch between the optical centers of the user's eyes, the cameras and the lenses, however, will still be present unless the VST HMD is orthostereoscopic.

In [90], a VST HMD is considered to be orthostereoscopic if the following conditions are met: the center of projection and optical axes of the cameras used to capture the video feed must coincide with those of the displays and the user's eyes; the distances between cameras, lenses, and the observer's eyes must be equal (IPD), and the field of view (FOV) of the cameras must be the same as the displays.

To mechanically respect these conditions, both the cameras and the lenses would have to converge to the focus point (toed-in HMD). Currently, commercial VST HMDs use a parallel setup instead, with fixed cameras and lenses as to obtain an orthoscopic HMD the cameras and the lenses would need to be motorized and move coherently with the user's eye movement, which is mechanically challenging. This leads to aberrations (e.g. diplopia) that can potentially hinder the interaction [2, 90, 99]. There are studies aimed at developing orthoscopic VST HMDs [13, 31, 86], or software solutions such as [19], where it is shown that it is possible to use parallel displays to create quasi-orthoscopic HMDs, reducing the perspective distortions caused by the parallax by appropriately warping the camera frames with a perspective transformation. This study, however, will focus on commercially available HMDs. To the author's knowledge, a true orthoscopic view (thus, without geometrical aberrations) is currently unavailable in all commercially available HMDs.

2.2 Perception, Interaction and User Experience in AR

Several studies already reported an underestimation in perceived egocentric distances in VR scenes (see [77] for a review). Since there are several possible causes (which also have interactions among them), the phenomenon is still not yet fully understood [87]. Moreover, in [36] the authors compared different VR and AR situations with and without an HMD by reporting an underestimation in all conditions, though the magnitude of the errors varied substantially. What we currently know is that the human visual system estimates depth on the basis of several *depth cues*, which can interact each other to disambiguate percepts depending on the situation. The same problems that arise with VR HMDs also applies for VST HMDs, as some depth cues are often hindered or altogether negated in a similar way. OST HMDs also do suffer from the lack of some cues, and the effects of distortions caused by the optical system.

To measure the user's perception, some kind of interaction task is often required. In the context of VST/OST HMD comparison, several studies have been performed using different interaction tasks. In [24], the authors compared the two HMD technologies in the context of text reading, finding that OST seems to be better for text readability. In [64], the combined effects of two types of latency in AR (real-world latency and virtual object latency) are analyzed, although with an AR simulation approach (i.e. the real and augmented portions of the AR training scenarios are simulated in VR). The results confirmed that latency could play a role in AR applications, and it is something researchers have to take into account. Other specific comparisons between VST and OST devices are about the vergence-accommodation conflict [54] or the sense of presence for phobia treatment to small animals [49].

Generally speaking, the type of interaction can potentially introduce a bias in spatial estimation judgements, due to several cognitive factors. According to the Human Processor Model (MHP) by K.Card et al. [18], the human memory is split in several interlinked storages (see Figure 5). The perceived stimuli are first stored in short term memory, which has a quick decay rate (i.e., is quickly erased), and acts like a buffer (or a RAM). Stimuli perceived through different senses have different decay rates, i.e. visual short term memory decays faster than auditory. Interestingly, the senses which store information which typically requires more memory to be digitally stored (e.g. images/video) have quicker decay rates than those which encode simpler information (e.g. audio), suggesting that the decay rate is closely related to the buffer's size. Remembering a stimuli for longer than its associated decay time is only possible if the cognitive subsystem is activated, which implies an increased cognitive load.

Through rehearsal and drawing from the limited reserve of active attention, the stimuli can then be encoded into long term memory, which acts like the human version of a hard drive and can be split in several subtypes (e.g. episodic, procedural, semantic, etc.). Long term memory still decays over time if the rehearsal process is interrupted, and can be warped on each iteration, leading to false memories. Similarly to its digital counterpart, fetching information from long term memory is a slower process with respect to short term memory, and the data encoding process (e.g. the "writing to disk" process) is subject to interferences and errors. For example, according to Jost's law of forgetting [47], if 2 memories are of the same strength but different ages, the older will decay more slowly than the younger.

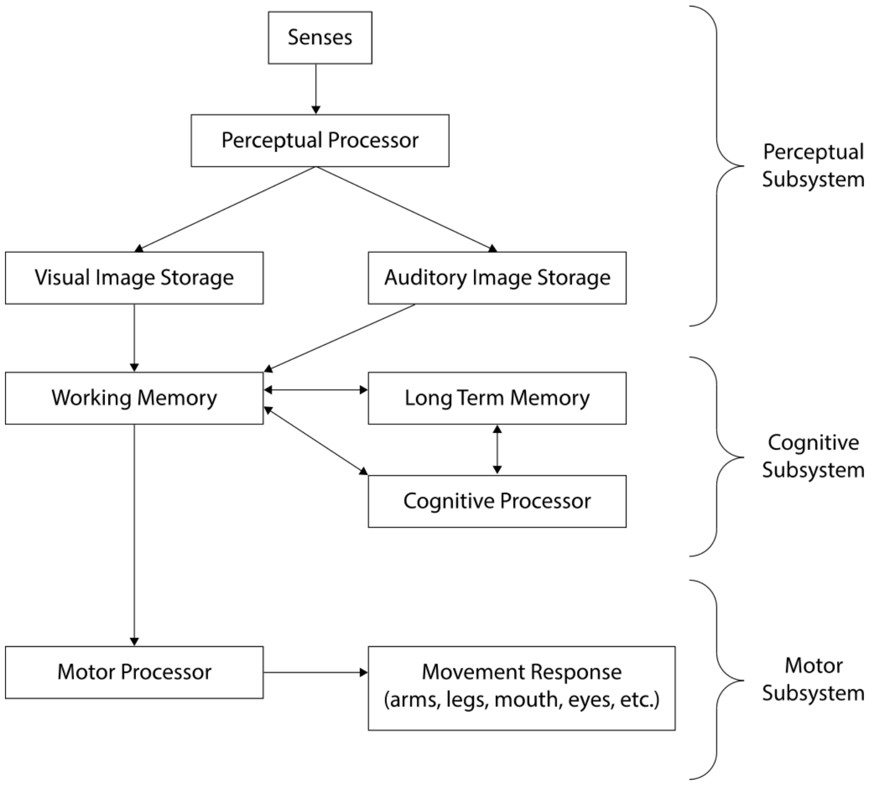


Figure 5: The Human Perception Model by K.Card et al. [18]. The perceived stimuli are stored in the short term memory present in the Perceptual Subsystem (Visual Image Storage and Auditory Image Storage). Through the working memory, the stimuli can then be stored into Long Term Memory, but the process requires rehearsals and active attention, a limited cognitive resource.

The decay half-life of visual image storage is between 90 and 1000 milliseconds, and is able to memorize a limited amount of items. The general consensus [61] indicates 7 ± 2 items, which can be increased by *chunking*, e.g. trying to remember a 9-digit number is easier if instead of memorizing single digits the number is split into three triple-digits numbers. Given that short introduction on the human perception model, it is clear that whenever the perceived stimulus is memorized in short-term memory, the judgement will be increasingly affected by memory alteration or erasure as the time required for the interaction increases. For this reason different interaction protocols will be discussed in this chapter.

In the next section, the depth cues theory for spatial perception will be discussed in detail. The subsequent section will make an overview on the interaction techniques used to evaluate the user's perception, while the final section of this chapter will discuss the user experience (UX) design implications behind AR applications.

2.2.1 *Depth Cues Theory*

In the past years, several researchers have addressed the problem of evaluating such devices' perceptual issues. In [55], the authors provide a classification of perceptual issues in augmented reality by considering the advantages and disadvantages of the different technologies. Depth ordering and perception are some of the most important issues since depth distortion primarily affects interaction.

Between the ten depth cues that are recognized to be more important in the depth estimation process [23, 42, 87] the most prominent in peripersonal space are binocular disparity, binocular convergence/accommodative focus, relative size, and occlusions.

To convey binocular disparity, AR HMDs have to render different views for each eye, which are approximately spaced by an interpupillary distance (IPD) of 63 mm [26]. However, the exact positions of the virtual cameras that render the separate views for each eye change depending on the user. As we saw in 2.1.2, commercially available VST HMDs use a parallel setup, with fixed cameras and lenses, which introduces aberrations such as diplopic vision [19, 90]. Of course, monocular HMDs such as the Google Cardboard are unable to provide the binocular disparity cue: the video feed from the frontal camera of the smartphone is simply replicated for both eyes. Similar problematics are present in OST HMDs to a lesser extent, as to achieve locational realism, the position of the user's eye must be obtained [37]: since eye trackers are still rare in commercial HMDs, calibration through alignment tasks is usually required. OST HMDs do not remove this cue for real objects, as users can see through the transparent lenses, which eventually only add a small distortion due to the light refraction.

Binocular convergence and accommodative focus, in a real world scenario, are usually tied together in a single accommodation-convergence reflex. Like in the previous case, the cardboard-based VST mostly negates the contribution of these cues, as the user must keep focus on the same image, which is focused depending on the camera settings. Also in this case, the OST HMD should perform better for real objects.

The most important monocular depth cue in our context is arguably the perspective of the scene, and consequently the relative size of objects. It is demonstrated that if the real size of an object is known, an assumption is also made on its distance depending on the perceived size. The cardboard-based HMD slightly shrinks the captured image, thus partially distorting the contribution of this depth cue.

Occlusions do have a great importance in depth estimation, but in open-loop judgment task they can only provide information about the relative positions of objects in the scene, as motion parallax, not their absolute egocentric distance. Many other depth cues, such as atmospheric haze, shading, texture gradient and height in the visual field, are either not applicable in peripersonal space or are limited in current AR HMD devices.

In a former study [20], a compression of perceived depth when using a VST HMD for AR was experienced, mostly due to the mismatch between the smartphone focal length and the observers' one. Errors (variable with respect to the distance) in visually perceived distance judgment tasks, by considering VR and real world in [67], and by considering a tablet-AR based walking experiment in [88], are also reported. A comparison of closed-loop and open-loop near-field distance perception in AR is reported in [85], by showing that blind reaching was significantly underestimated.

In a real-world scenario, binocular convergence and accommodative focus are usually tied together in a single accommodation-convergence reflex, which is problematic to preserve in VST HMDs, due to the conflict arising from the presence of the display focal plane at a fixed distance from the eyes.

Perspective is also a strong monocular depth cue in peripersonal space, as well as the relative size of observed objects. When familiar objects (thus, with an approximate known size) are observed, the distance is also estimated depending on the perceived size. Some famous optical illusions exploit this phenomenon by resizing common objects to conceal the object's real distance.

Finally, occlusions and motion parallax [70] provide a strong cue regarding the object's relative positions of objects in the scene.

Several other cues, such as atmospheric haze, shading, texture gradient, aerial perspective, dimensionality (i.e., 2D vs. 3D shapes), and height in the visual field, are either more related to medium/large distances or currently not conveyed through commercially available AR HMDs (e.g., due to limited FOV or resolution). In [25, 81] the impact of some of these cues on spatial perception is discussed, and a depth underestimation is always present, suggesting the HMD is not able to properly convey all the depth cues.

Having established that both OST and VST devices should be calibrated to be used and that such a calibration should take into consideration human perception, a still open question is whether there are relevant perceptual differences between the two technologies and whether such differences affect interaction, especially in the peripersonal space, i.e., at near reachable distances, in the context of egocentric interaction. In the literature, the tradeoffs between optical and video see-through HMDs with respect to technological, perceptual, and human factors issues have been largely debated [79, 80].

2.2.2 *Interaction in AR*

As already introduced, when performing perceptual experiments, the type of interaction used to express the spatial perception judgement can introduce bias into the judgement itself. Over the years, several types of interaction have been used to judge spatial perception.

It is necessary to make a first distinction, to obtain three interaction zones [23, 65, 66, 73] which are related to the distance with respect to the user: *peripersonal space*, *extrapersonal space* and *vista space*. With *Peripersonal space* we refer to the space in close proximity to the user (up to about 1.5m), which is commonly used in reaching, grasping and manipulation tasks. AR perception in extrapersonal space (medium distances, up to about 30m) and vista space (large distances) related to respectively motion, navigation and general context/orientation, is currently less explored, as most AR applications (with the exception of handheld AR, e.g. [88]) are often indoor and at close distances.

A second distinction can be made on the presence or absence of a physical interaction, i.e., the judgement can be expressed either by verbal estimation or by physical matching, for example by performing a manual alignment task.

Finally, we can split between open-loop and closed-loop interactions, depending on the presence or absence of a control feedback provided to the user. Below, the most widespread techniques to register the interaction used to express the perceived spatial perception are briefly introduced.

In blind interactions, a feedback is not provided to the user, which thus performs the movement control during the judgement in open-loop. Depending on the interaction zones, it may involve blind walking (for extrapersonal space), i.e. walking up to the perceived location of an AR generated stimulus, or blind reaching/pointing (for peripersonal space). In blind tasks, the stimulus is first displayed for a set amount of time, and then disappears before the user performs the physical alignment, either by walking or by pointing/-grasping. As we pointed out earlier, the visual short term memory is volatile, thus absolute depth distance judgements tend to be more imprecise as the time used to perform the interaction increases. With absolute judgements we refer to judgements of perceived locations which cannot be inferred from the geometrical structure of the scene or the relative position of its components, i.e. there must be no elements which could be used to remember the perceived stimuli location by linking it to a nearby close persistent object.

In perceptual matching interactions, on the other hand, the user is allowed to align himself with the perceived stimuli in closed-loop. In this scenario, the user is able to perform fine adjustments until a perfect overlap is perceived, thus the noise potentially introduced by memory degradation is isolated even if the interaction requires an extended amount of time.

Although absolute depth distance judgments tend to be metrically imprecise as the distance increases [87], relative depth perception (e.g., to discern which object is closer between two distinct ones) is known to be much more accurate. Relative depth perception can also be performed either in a blind setting or not. In this type of setup, the scene usually contains a visible structure which can be used as reference, which can be as simple as several items scattered on a table.

In [87], the authors review and discuss protocols for measuring egocentric depth judgments in both VR and AR and the well-known problem of depth underestimation in virtual environments. Among the first authors who tried to evaluate depth perception, they considered three different protocols: perceptual matching, blind walking, and verbal estimation. Their work focused on depth perception at medium and far-field distances, which are important distances for several compelling AR applications, but not for peripersonal tasks, which are the focus of this study.

Interestingly, they found evidence for a switch in bias, from underestimating to overestimating distance, at about 23 meters (see also [89]). In the blind walking tasks, they found that AR objects' egocentric depth is underestimated, but to a lesser degree than previously observed in VR. It is worth noting that underestimating distances is also reported in natural viewing (or VR or AR), e.g., in [100] the authors reported a good evaluation for distances only up to 40 cm, by considering an alignment task. In real-world scenarios, in [62] the authors analyzed distance perception with a pointing task, finding out that accommodation can act as a source of ordinal distance information in the absence of other distance cues. Finally, in [46], the authors quantified egocentric perception, but they considered a walking task, with distances from 2 to 8 meters from the observer.

2.2.3 *User Experience*

User experience and usability goals (i.e. goals reachability, efficiency, satisfaction) are an important aspect of every interface design and a central discussion topic for human computer interaction, and AR interfaces are not an exception.

One primary goal of user interface design is to reach certain learning goals, i.e. sufficient learnability, memorability and safety. Learnability refers to the difficulty of the training phase required to become proficient with the given interface: depending on the application, the learning curves assume different shapes. An interface aimed at professional use tends to have a steeper/logistic learning curve as there are more functions to learn, while an interface oriented at casual use should require the least effort possible to learn the most basic functions. Memorability refers to the difficulty of remembering the training over time (i.e. retain the proficiency), and is closely related both to the complexity of the system (and thus the length of the training phase) and to the mediums leveraged during the training (e.g. sounds are remembered more than visuals). Safety refers to the possibility of recovering from user's error.

One of the biggest strengths of AR interfaces is the opportunity of decreasing the user's cognitive load and task interaction cost, by offering a more naturalistic interaction approach with respect to cluttered interfaces with many gizmos and menus. Indeed, the human's brain has adapted through millenniums of evolution to interact with its environment through manual interaction and tools. According to the common coding theory, perceptual and motor representations are closely interlinked, and activating one event triggers the other and vice versa. As an example, when we look at an object, the brain automatically computes possible interaction routes and grasping positions: this phenomena is caused by the past ideomotor learning, i.e., remembering the consequence of a specific set of movements. The ideomotor theory [44, 45, 74], fueled by the discovery of the mirror neurons [32, 78] present in the premotor cortex (F5), suggests that this link between ideomotor learning and prediction works similarly to forward and inverse kinematics computations of control theories.

Having established that the human brain is deeply adapted to physically interact with the environment, AR has the advantage of removing abstraction from current computer interfaces, often based on the WIMP paradigm (i.e. Windows, Icons, Menus, Pointing device), resulting in a more intuitive interaction. According to the Khaneman theory [50], our cognitive system is based on intuition and reasoning. Intuition is triggered to a subconscious level by perceived object's characteristics, while reasoning requires a voluntary action which requires both working and long-term memory and a sufficient attention. The presence of two separate cognitive systems is manifested by the dichotomy between knowing something (theory) and being able to do it (practice). For example, we can perform very complex tasks without being able to explain them (walking, running, balancing) or know every detail of a procedure but not being able to put it in practice, due to the lack of muscle memory. Actions triggered by the intuition are much quicker to compute and have a much lower cognitive load than reasoning-based ones.

This further explains the potentiality of AR interfaces. For example, in a simple raster graphics editor, on a traditional interface we can expect to see a menu with different stylized icons (e.g. pencil, lazo, magic wand, eraser): even if those icons are mostly standardized across similar other softwares, their affordance level is still closely related to the familiarity of the user with similar systems. In an AR raster graphic editor, those tools can be represented by actual physical items (e.g. a physical pencil or eraser) which are more likely to be part of past experiences and also trigger ideomotor learning, thus resulting more intuitive.

We can thus assume that if an AR experience manages to seamlessly integrate virtual and real content, the obtained interface would satisfy most usability goals. One key problem, however, is the enabling technology, i.e. the HMDs. As we saw in previous sections, achieving locational realism and photorealism is not a trivial task. The presence of parallaxes, distortions, latencies, the approximation of the models all lead to perceptual mismatches. The HMDs themselves are often cumbersome and convey a varying amount of light directly to the user's eyes. The calibration procedures can often be long and tedious.

For these reasons, a few aspects can be taken in consideration to measure the user's experience with different HMDs: the cybersickness, the sense of presence, and the cognitive load.

Cybersickness is an index that has started to be measured for VR systems, but due to the similarities of the optics and ergonomics of VR and AR HMDs, the same principles apply for AR. It can be assessed through self-evaluation questionnaires or through the analysis of physiological data (e.g. hearth rate, eye blink rate, skin conductance response, EEG delta wave). Cybersickness can be defined as the potential development of several disorders such as nausea, fatigue, focusing difficulties, eyestrain and other general discomfort symptoms while or after being subject to visual stimulation. It is caused by several factors, such as the mismatch between the visual information perceived and the motion information inferred by the vestibular system (which is notoriously evident in roller coaster simulations in VR), and the presence of latencies which desynchronize the movement of the scene with the expected movement inferred by the self perception of the head's movements (kinethesis). Even when not directly causing sickness symptoms, latencies severely impair proper interaction, e.g. in [48] it is shown that even a 10ms latency reduces touch performance in direct-touch pointing tasks.

Past studies have proposed several self-assessment questionnaires to measure cybersickness. One of the most widespread questionnaires for self-assessment of cybersickness is the Simulator Sickness Questionnaire (SSQ) [101], which splits the developed symptoms into four categories: nausea, oculomotor, disorientation and total score. The questionnaire is composed by two sections to be filled before the use of the HMD (to define a baseline of the user's conditions) and afterwards.

The sense of presence, in MR contexts, measures how much the virtual content is perceived as spatially authentic. Due to its subjective nature, measuring the sense of presence through physiological data is difficult, as commonly measured indexes (e.g. hearth rate or skin conductance response) would be difficult to directly relate to the sense of presence experienced, if any sway can be detected at all. Most of the time the sense of presence is thus measured through self-assessment questionnaires, like the IGroup Presence Questionnaire (IPQ) [82], which is based on a set of multiple choice questions split into three macrogroups (spatial presence, involvement, and experienced realism). The number of self-assessment questionnaires specifically targeted at measuring the sense of presence in AR environments (e.g. [75]), however, is still quite low. Most of these questionnaires are targeted towards purely virtual environments (VR), thus some of the questions can be more difficult to interpret when transposed to AR. For example, in the IPQ questionnaire, one of the questions asks if *the virtual world seemed more realistic than the virtual world*, i.e., if (while in VR) the user forgets about the real surrounding environment in favor for the virtual one. in AR, however, both the virtual and real world will be present at the same time, and the question becomes harder to interpret: the natural answer will always disagree, as the augmentation will always be evident or, at most, merged with the real environment, but the spatial awareness of the real surroundings will never be lost.

2.3 Registration

With registration (or image registration) we refer to the process of transforming heterogeneous data into one single common coordinate system. The data may be heterogeneous for several reasons, such as being obtained from different tracking devices (e.g. RGB/RGB-D cameras, IR, sonar, etc), from different viewpoints, or at different time frames [14]. Generally speaking, for two sets of data to be compared together, they must be expressed with respect to the same reference frame (i.e. "observer"), where time can also be considered one of the coordinates of the frame. When registering heterogeneous data, we define the position and orientation of a *global (or world) reference frame* and convert all other measures to be expressed with respect to the it. The global reference frame is usually positioned in such a way that measures expressed in its frame are easily interpreted, e.g. for a manipulator robot it could be positioned on the end effector, or onto the base, to easily interpret the relative position of graspable objects in front of the robot, rather than on a random joint of the arm.

The next sections will briefly introduce to the problematics related to the registration of the different devices that are commonly present into an AR setup, and the registration of the user's interaction with the system.

2.3.1 *Multi-device registration*

A common scenario in AR is to have several sets of data coming from different sources and needing to visualize it in a common shared system. In a linear setting, to transform a point ${}^{a}p$ expressed e.g. in a reference frame A to a different reference frame B it is required to know the transformation matrix ${}^{B}_{A}\mathbf{T}$, which is usually expressed either as the combination (Eq.1) of the rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ (Eq. 2) which align the reference frame A on top of the reference frame B or as homogeneous matrix which represents a transformation which encapsulates a rotation followed by a translation (Eq. 3).

$$ {}^{B}_{A}\mathbf{T} = [\mathbf{R} \mid \mathbf{t}] \tag{1} $$

$$ \mathbf{R} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \qquad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{2} $$

$$ {}^{B}_{A}\mathbf{T} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

If the transformation matrix is expressed as in eq. 3, the point ${}^{a}p$ expressed with respect to the reference frame A can be converted to be expressed in relation to the reference frame B by simply multiplying its coordinates for the transformation matrix as displayed in eq. 4

$$^{b}p =^{B}_{A} \mathbf{T} \quad ^{a}p \tag{4}$$

The opposite transformation, i.e. to transform a point ^{b}p expressed in the reference frame B to be expressed in the reference frame A, is simply the inverse of the transformation matrix $^{B}_{A}\mathbf{T}$, which is denoted as $^{B}_{A}\mathbf{T}^{-1}$ (see Eq.5).

$$^{a}p =^{B}_{A} \mathbf{T}^{-1} \quad ^{b}p \tag{5}$$

Since the transformation matrix $^{B}_{A}\mathbf{T}$ is a block diagonal matrix, it is always invertible. The inverse of the rotation matrix $\mathbf{R}$ (which always have a matrix determinant equal to one) is simply the transpose, i.e. $\mathbf{R}^{-1} = \mathbf{R}^{T}$.

In reality, we have also to deal with non-linearities, i.e. we must also take in account non-linear distortions present in the acquisition(s) system (e.g. tangential or radial distortions introduced by the camera lenses). Those non-linearities must also be modeled to rectify the system back to an approximate linear case.

If the transformation matrices between the two frames are unknown, they can be derived through a calibration process, which requires the knowledge of the positions of a set of 3D points in both reference frames.

2.3.2 *User registration*

Interaction is a key aspect of AR experiences, and over the years several technologies have been developed to enable different interaction techniques to interface with the virtual content. To begin with, we can make a first distinction between *direct* and *indirect* interaction techniques, which mirrors the definition of direct and indirect locator devices for traditional computer interfaces [29].

In an *indirect* interaction, the effect or consequence of the user's movement is observed in a different location with respect to the original movement. This is the case, for example, when interacting through the operation of a keyboard or a controller of unknown 3D position (e.g. a joypad/joystick).

In a *direct* interaction, the effect of the user's movement takes place in the same location of the triggering movement. An example is direct hand manipulation (through hand tracking) or gaze-based interaction.

It must be noted that *indirect* interaction is always *mediated*, in the sense that it requires a process of eye-hand (or body) coordination, thus it always bring a bigger cognitive load with all the derived consequences: slower response time, higher failure rate, lower affordance, higher user's strain.

We can then differentiate depending on the amount of data which needs to be processed at each time instant. Technologies which requires additional hardware to be worn in addition to the HMD (wearables, or tool-based) usually provide a very small stream of stable data (e.g. single keystrokes or button triggers), while data-hungry methods (such as vision based ones) are much more computationally heavy but usually do not add additional burdens to the user, either thanks to external sensors (e.g. an external RGB-D sensor which records the user's movements) or by using the ones embedded in the HMD (e.g. cameras).

Wearable and tool-based technologies (e.g. keyboards, tracked controllers, or smart bracelets) usually have a higher consistency, i.e. the outcome of an action has a much higher predictability than vision-based ones, since the obtained measures are less subject to noise. Their indirect nature lead to less natural interactions, as the user needs some training time to learn the outcome of each input given.

Vision-based methods (e.g. hand pose estimation and tracking) are computationally heavier, but require a cheaper array of sensors, do not restrain the user's movements, and can potentially estimate high DoF kinematic models (e.g. [16, 17, 84, 103]). The enabled interaction (e.g. bare hand interaction) is more natural, but low tracking stability and the lack of a haptic feedback can greatly impact the predictability of the system outcome. Indeed, in [11] the authors found that people preferred to use controllers instead of a hands-free interaction based on a Leap Motion tracking device, due to the better system stability. The given interpretation is that since nowadays we are very familiar with the daily use of tools, machines, keyboards and controllers, the small increased overhead needed to learn how to use a simple controller is vastly repaid by the increased system stability. This may no longer be true on more complex scenarios, however.

In Figure 6 a representation of most widespread technologies used to register the user's interaction is displayed, approximately arranged according to the previous mentioned distinctions. The complexity of the system grows as any of the two axes grow, both due to the increasingly difficult task of interpreting the user's actions from increasingly more complex types of data (e.g. button presses against EMG signals) and also due to the increased amount of raw data which needs to be processed in real-time.

2.4 Device Calibration

Generally, to limit the effects of the geometrical distortions caused by the HMD, a calibration is required. Calibration estimates the camera's parameters and the object models to match the virtual objects with their physical counterparts. VST and OST devices are composed of different systems (to track the real-world environment, the users' head pose, and to display the augmented contents) with different reference frames that must be aligned. The calibration parameters are the optical characteristics of the physical camera as well as the position and orientation of various entities such as the camera, the trackers, and the various objects [33].

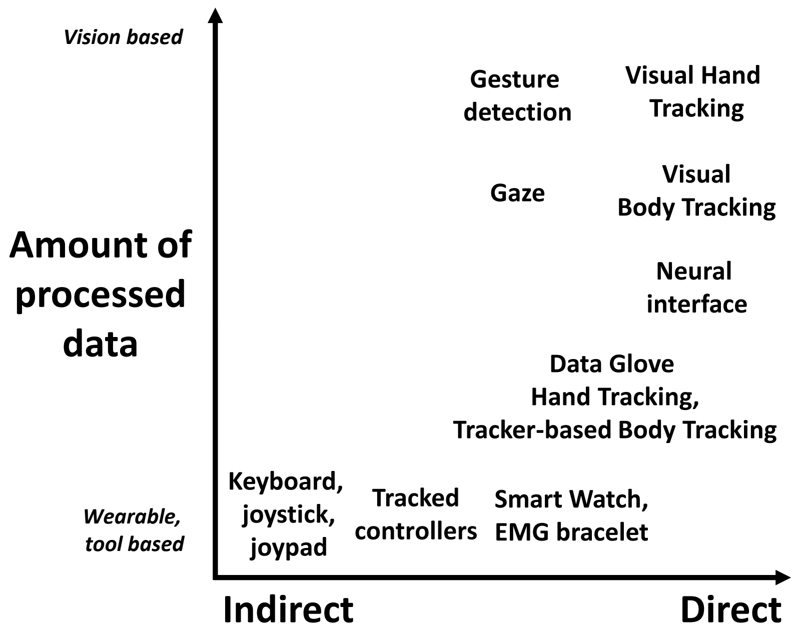


Figure 6: Different technologies used for interaction in AR environments. Wearable or tool-based technologies require the user to interface/wear additional hardware. Externally tracked ones are less invasive to the user, and exploit either the HMD cameras or external sensors. Indirect methods decouple the input interface from the spatial occurrence of the outcome, while direct methods closely resemble interaction with real objects.

2.4.1 *OST Calibration*

With VST HMDs, the problem of finding the pose of the camera which captures the video stream can be solved by using the same camera to track the movements of the HMD (e.g.[51]), which is usually what happen with current VST HMD setups (e.g. ZED mini). With OST HMDs, the virtual camera position is represented by the user's eyes, thus a calibration to obtain the eye-display transformation is required. A recent survey [37] describes several OST devices calibration methods, classifying them into manual, semi-automatic, and automatic approaches.

Manual calibration methods require the test subject to perform a manual alignment task, which is needed to compute all the required parameters that define the projective geometry of the user-HMD system. The problem of manual calibration methods is that if the HMD slips in a different position than the one used during the calibration procedure, the obtained parameters are no longer valid.

Semi-Automatic methods seek to simplify the calibration process by reducing the number of alignments required. This is normally achieved by computing only the parameters that change between different users (e.g. [71]) or between different sessions for the same user (e.g. [34, 35]). Semi-Automatic methods seek to reduce the reprojection error by being less reliant on the user precision during the alignment task of the calibration. It must be noted that implementing these calibration techniques is usually more elaborate with respect to traditional Manual methods, and sometimes requires additional hardware. As an example, in [71] the parameters were split between the eye model and the display model, which was calibrated separately using a mechanical apparatus.

Automatic calibration methods do not require any user input, as they are able to obtain the 6-DoF pose of the eyes automatically, mostly by using eye tracker sensors. Of course, Automatic calibration represents the best option when available, as it accommodates for changes of the geometry during run time (although with a processing overhead). The presence of an eye tracker integrated in the HMD also enables enhanced gaze interactions (as opposed to the fixed crosshair-centered based ones) and more realistic rendering techniques (e.g. foveated rendering). Currently, however, most commercial HMDs do not provide integrated eye tracking functions, thus implementing Automatic calibration methods can be troublesome due to the difficulties related with detecting eye movements using external sensors. For this reason, these methods will not be covered in this study.

Many approaches have been proposed and developed for OST calibration, starting from considering that a non-accurate calibration hampers users' ability to solve tasks in the AR environment [68, 91]. It is interesting to note that human factors contribute to a significant extent to calibration errors. Even different calibration procedures for the same calibration algorithm yield significantly different accuracy [4]. A study comparing different confirmation methods for the alignment procedure (keyboard button press, handheld device, voice activation, and gaze-based or wait-based) show that lower movement inducing methods lead to a lower reprojection error [58], suggesting that even minimal sway significantly impacts the calibration process.

The evaluation of methods to calibrate OST HMD devices is often performed with user studies, e.g., [38, 63], both with quantitative [59] and qualitative feedbacks.

2.4.2 *VST Calibration*

There are several studies that aim at reducing geometrical aberrations present in VST HMD devices through calibration or software solutions [15, 19, 56, 57, 72]. The calibration requirements of a VST AR system have also been described in [95, 97], although with a monitor setup. One of the main problems with interacting in AR environments experienced through VST HMD devices is that the human eye and the camera's intrinsic parameters are different, thus hampering a correct estimation of egocentric distances [10, 21].

As described in section 2.1.2 the parallaxes present in non-orthoscopic VST HMDs also introduce several aberrations such as scale factors and distortions of horizontal/vertical disparities which is only partially adjustable through software solutions [19]. Moreover, the HMD lenses introduce non-linear distortions (e.g. tangential or barrel), which in are compensated by undistorting the cameras frames and using a pre-distortion function before reprojecting them onto the HMD display [19, 56, 57, 72]. The problem of geometric consistency between displayed images and real scenes in AR is specifically addressed by using a VST handled display or tablet [92]. In that paper, the authors tested the approach (to approximate user-perspective images rendered by homography transformation of camera images) with a user study, where people was asked to rotate a virtual cube to match the pose of a real one.

3 The Registration Framework

This chapter describes the structure of the registration framework designed and developed[1] to bring all the used AR devices and tracking systems in the same reference frame.

The goal is to have all the devices share the same tracking system, thus the same reference frame, to coherently compare obtained data. Since the considered tracking system can track any object's pose using trackers, we convert all other devices coordinates by finding their relative transformations with respect to the tracker rigidly attached to each device.

Our setup is composed of two AR HMDs, one VST, and one OST. The OST is the Meta2 (Metavision), while the VST is the HTC Vive Pro equipped with ZED mini stereo-cameras. The OST HMD field of view (FoV) is 90° and has a Quad-HD (QHD) resolution (split between the two eyes), thus 2560×1440 pixels in a 16:9 aspect ratio. The Vive Pro HMD, on the other hand, has 110° FoV and a resolution of 2880x1600 pixels (1440x1600 per eye). The Vive Pro, while used as an AR device, is further restrained by the ZED Mini used for the see-through, which has a maximum resolution of 2560x720 pixels and a FoV of 90° (HxV). The lighthouse-based tracking system of the HTC Vive Pro has been used for both HMDs, due to its precision [12, 69] and availability.

To track the user finger during the reaching task, a Kinect V2 RGB-D sensor was used.

Unity 3D (version 2017.4.17f1) was used as the graphic engine. The framework has been tested on a PC with the following specifications: GPU NVIDIA GeForce GTX 1080, processor Intel(R) Core(TM) i7-8700 @ 3.20 GHz, 32 GB of RAM, and as an operating system Windows 10 Pro 64 bit.

The proposed registration framework is composed of several modules (Figure 7). All the devices are registered in the HTC Vive reference frame, for several reasons. First, the Lighthouse tracking system is precise, robust to occlusion and works in the darkness. The world origin is placed in a controlled, repeatable position across multiple sessions, in the center of the rectangular area set during the steamVR room calibration. The availability of Vive Trackers, which are already tracked in the HTC Vive ecosystem, enables the registration

1 The framework repository is available on GitHub [6], distributed under *the Unlicense* [1] public domain equivalent license. The documentation is provided in the Appendix.

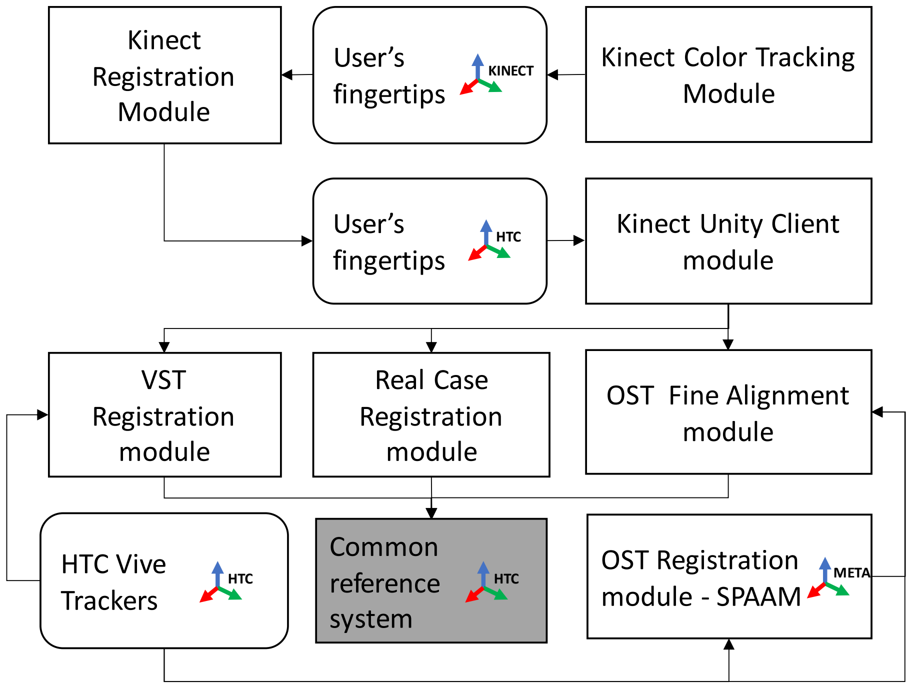

Kinect Registration Module
User's fingertips
KINECT
Kinect Color Tracking Module
User's fingertips
HTC
Kinect Unity Client module
VST Registration module
Real Case Registration module
OST Fine Alignment module
HTC Vive Trackers
HTC
Common reference system
HTC
OST Registration module - SPAAM
META

Figure 7: The flowchart encapsulates all the modules of the proposed registration framework, used during the design of the experiments described in chapters 4.2 and 4.3. Rounded-edge blocks represent 3D points, tracked with respect to the displayed reference frame and registered in the connected systems.

of any other device, assuming a rigid transformation between the tracker and the given device is obtainable. Finally, the HTC Vive is much more economical than most motion capture (MOCAP) systems. Without loss of generality, we assume the proposed approach to be valid under any other choice of visualization devices and tracking systems. Most procedures have been carried out by using the raw data streams from the different sensors.

The OST Registration Module handles the registration of the OST HMD in the VST HMD reference frame, by means of a Vive tracker rigidly attached to the HMD with a 3D printed part. The rigid transformation between the Vive tracker and the OST HMD inner tracking system origin is first obtained with several iterations of the Single Point Active Alignment Method (SPAAM) [96], which gives a first estimation. Each user then performs an alignment task to deal with the residual alignment error. To avoid conflicts between the OST HMD SDK and the steamVR SDK, the HTC Vive measures are collected externally in a separate server program and sent to the client through an UDP connection.

In our case, the VST HMD was the HTC Vive, which is already tracked in the Lighthouse tracking system. The VST Registration thus just involves the calibration of the stereo cameras used for the video-see through. The registration of any other VST HMD in the HTC Vive system can be performed with the same approach used in the OST HMD. A custom mount to find, with a

Vive Tracker, the exact position on the HMD of the ZED mini used for the video-see through is provided as example.

The Kinect Color Tracking and Registration Modules handle the tracking of the user fingertip by means of a color filtering. The 3D position of the fingertip is then converted from the Kinect reference frame into the HTC Vive reference frame and integrated into Unity through an UDP socket connection.

The framework also includes a Real Case Registration module, which can be used to display visual stimuli in the real world in such a way that (i) the displayed stimuli are as coherent as possible as the AR case and (ii) no haptic feedback is provided if the experiment also involves a reaching task. The structure used is registered in the HTC Vive reference system, allowing the comparison of the perceived positions of the stimuli (obtained through the Kinect Tracking and Registration modules) and the real positions displayed (which are in a known position).

3.0.1 *OST Registration Module*

The Meta2 6DOF pose in the scene can be obtained through its internal SLAM (Simultaneous Localization And Mapping), which upon initialization asks the user to perform a short environmental scan by turning the head to collect enough features to build a 3D mesh of the surroundings through its integrated RGB-D sensors. This feature has been disabled and replaced by the HTC Vive Pro tracking system, using one Vive Tracker. This step is required as the Meta2 SLAM system places the origin of the world reference frame in the first pose registered after initializing. Thus it would be impossible to render objects in the same position with respect to the real world and compare the collected data among different users. Thus, the Meta2 HMD serves merely as a visualization device, as no advanced features of the Meta SDK were used. Since the rendering of the two eyes view is based on the position of the SLAM localization system reference frame, the HMD needs to be calibrated to obtain the transformation between the Vive Tracker reference frame (used to track the HMD) and its default localization reference frame. A screw rigidly fastens the HMD Vive tracker to the HMD through a 3D printed support (Figure 8 left), specifically designed for the Meta2 HMD. Moreover, each user has to calibrate the HMD each time it is worn, as for a proper AR rendering, the position of the user's eyes must coincide with the position of the virtual cameras in the graphic engine, and the intrinsic parameters of the eye-HMD model must also be defined.

3.0.1.1 *SPAAM*

To perform the OST HMD calibration, we used the popular SPAAM technique [96], as the Meta HMD has no eye-tracking sensors to be used for automatic calibration methods. For more insight on the SPAAM technique, refer to [7, 96], since we aim to provide a registration framework that allows having a common reference frame for different devices and real objects, not to devise a new calibration procedure for OST HMDs. Independent of the type of calibration, the purpose is to find the transformation matrix that can map HMD pixel coordinates to the user's eyes' coordinates.

The Vive tracker attached to the OST HMD will be referred to as the *mark* from now on for coherence in our setup [96]; we will use the same notation as in [96], with a few modifications.

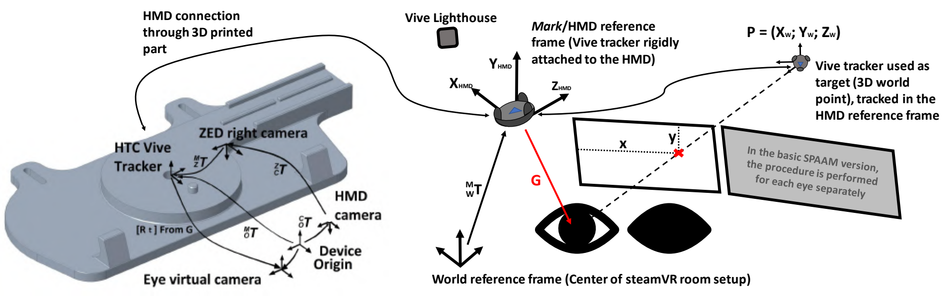

HMD connection through 3D printed part
HTC Vive Tracker
ZED right camera
$^M_Z T$
$^Z_C T$
HMD camera
$^C_O T$
$^M_O T$
[R t] From G
Device Origin
Eye virtual camera
Vive Lighthouse
Mark/HMD reference frame (Vive tracker rigidly attached to the HMD)
Y_{HMD}
X_{HMD}
Z_{HMD}
P = (X_W; Y_W; Z_W)
Vive tracker used as target (3D world point), tracked in the HMD reference frame
x
y
G
$^M_W T$
In the basic SPAAM version, the procedure is performed for each eye separately
World reference frame (Center of steamVR room setup)

Figure 8: The OST HMD registration setup. (left) The transformations needed to obtain a common reference frame with respect to the Vive lighthouses tracking system. Such transformations can be computed off-line, whereas the G matrix is obtained through the visual alignment task (right). The ZED camera has been temporarily placed on the 3D printed support only to find the $^Z_C\mathbf{T}$ fixed transformation, and then removed.

The OST SPAAM calibration technique is based on the pinhole camera model (Eq. 6) (see [27, 40, 105]), where the projective transformation **G** maps a 3D world point (x_w, y_w, z_w) into 2D pixel coordinates (u, v). The **G** matrix can be further decomposed as in Eq. 7, where **K** denotes the intrinsic parameters matrix, and $[\mathbf{R} \mid \mathbf{t}]$ defines the extrinsic parameters that describe the position and orientation (pose) of the camera.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{G}_{3\times4} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{6}$$

$$\mathbf{G}_{3\times4} = \mathbf{K}\,[\mathbf{R} \mid \mathbf{t}] \tag{7}$$

$$\mathbf{K} = \begin{bmatrix} f_u & s & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

The **A** system matrix (Eq. 9) which defines the projective transformation from world coordinates to virtual camera coordinates, where the virtual camera models the combined display system composed of the display and the human visual system, is formed of a 4×4 homogeneous transformation matrix ($^M_W\mathbf{T}$) which contains the pose of the *mark* expressed in world coordinates and a 3×4 projection matrix ($\mathbf{G}$) of the eye-tracker transformation (see Figure 8).

$$\mathbf{A} = \mathbf{G}\, {}^{M}_{W}\mathbf{T} \tag{9}$$

In our setup, the *mark* is already tracked in what are considered as world coordinates, thus, in this case, ${}^{M}_{W}\mathbf{T}$ represents what is referred to as **FC** in the original study. During the calibration, a crosshair is displayed (red cross on the left lens in Figure 8) and needs to be aligned to the fixed 3D point in the world, which in our case is tracked with a second Vive Tracker. To obtain the 3×4 matrix $\mathbf{G}$ there are 12 unknown parameters, but the projection matrix is defined up to a scale factor (λ), thus the number of independent parameters is reduced to 11. Since each alignment provides two equations, the number n of calibration points that needs to be measured during the alignment procedure must be at least 6. The 3D world points of each alignment i can first be brought in the *mark* (HMD) reference system. In this way the user's head movements are not restrained, as all the computations are carried out with respect to the current head pose. Hence, let the 3D world points of each alignment i have coordinates $P_{w,i} = [x_{w,i}, y_{w,i}, z_{w,i}]^T$, when expressed with respect to the *mark* (HMD) reference system. The pinhole camera model (Eq. 6) can then be written as in Eq. 10, where the homogeneous image coordinates $[u_i, v_i, w_i]^T$ of the projected point $P_{w,i}$ are related to the image coordinates $P_i = [x_i, y_i, 1]^T$ (i.e. the pixel coordinates on the HMD lenses) through the relation displayed in Eq.11.

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \mathbf{G}_{3\times 4} \begin{bmatrix} x_{w,i} \\ y_{w,i} \\ z_{w,i} \\ 1 \end{bmatrix} \qquad \text{for}\, i = 1, \cdots, n \tag{10}$$

$$\begin{aligned} x_i &= u_i / w_i \\ y_i &= v_i / w_i \end{aligned} \tag{11}$$

Assuming the matrix $\mathbf{G}$ has a structure as in Eq. 12, the projection model can be written as in Eq. 13.

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \end{bmatrix} \tag{12}$$

$$\begin{aligned} u_i &= g_{11}\, x_{w,i} + g_{12}\, y_{w,i} + g_{13}\, z_{w,i} + g_{14} \\ v_i &= g_{21}\, x_{w,i} + g_{22}\, y_{w,i} + g_{23}\, z_{w,i} + g_{24} \\ w_i &= g_{31}\, x_{w,i} + g_{32}\, y_{w,i} + g_{33}\, z_{w,i} + g_{34} \end{aligned} \tag{13}$$

By substituting the relation of Eq. 11 inside Eq. 13, the projection model becomes as in Eq. 14. The model can then be rearranged in function of the unknown parameter vector $\mathbf{p} = [g_{ij}]^t$ which contains all the twelve entries of the $\mathbf{G}$ matrix (Eq. 12) in a column vector.

$$\begin{aligned}
x_i(g_{31}\, x_{w,i} + g_{32}\, y_{w,i} + g_{33}\, z_{w,i} + g_{34}) = \\
g_{11}\, x_{w,i} + g_{12}\, y_{w,i} + g_{13}\, z_{w,i} + g_{14} \\
y_i(g_{31}\, x_{w,i} + g_{32}\, y_{w,i} + g_{33}\, z_{w,i} + g_{34}) = \\
g_{21}\, x_{w,i} + g_{22}\, y_{w,i} + g_{23}\, z_{w,i} + g_{24}
\end{aligned} \tag{14}$$

The final homogeneous equation to be solved thus takes the form of Eq. 15, where the matrix **B** is the matrix constructed by adding two rows for each alignment performed as shown in Eq. 16. The system can then be solved through singular value decomposition (SVD) of the matrix **B** ($\mathbf{B} = \mathbf{UDV}^T$) to extract the **G** matrix from the column of the **V** matrix which corresponds to the smallest singular value.

$$\mathbf{Bp} = 0 \tag{15}$$

$$\mathbf{B} = \begin{bmatrix}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_{w,i} & y_{w,i} & z_{w,i} & 1 & 0 & 0 & 0 & 0 & -x_i x_{w,i} & -x_i y_{w,i} & -x_i z_{w,i} & -x_i \\
0 & 0 & 0 & 0 & x_{w,i} & y_{w,i} & z_{w,i} & 1 & -y_i x_{w,i} & -y_i y_{w,i} & -y_i z_{w,i} & -y_i \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix} \tag{16}$$

In [96], the 3×4 projection matrix **G** (in Hartley and Zisserman [40] notation) was converted in the 4×4 projection notation used by OpenGL by pushing the parameters into a 4×4 orthographic projection (for more insight on the procedure, see [94, 96]). In our study, we dissected the camera matrix into its intrinsic and extrinsic parameters by RQ decomposition, and applied them separately to a standard camera object in Unity, by using its *transform* (for the extrinsic parameters) and *physical camera* (for the intrinsic) properties to render the virtual objects correctly.

This calibration method works in the monocular case and can be adapted to a stereoscopic model by aligning 3D objects instead of points [33]. For simplicity, we calibrate separately for each eye.

From preliminary testing, we observed that outliers caused by misalignments during the calibration (due to breathing and small head movements) could introduce a non-trivial amount of error, thus to reduce the variance between calibrations, we implemented a RANSAC procedure [28]. We thus collected $n = 15$ alignments for each eye between the virtual crosshair and the fixed 3D point instead of the minimum 6 alignments. An alignment was then considered an inlier if the reprojection error was under 0.1 mm. We computed the projected pixel size (0.059 mm) by considering the surface ratio between the projecting LCD surface and the lenses. The stopping criteria are based on the number of iterations i, which is updated every time a new model with more inliers m is found, based on the probability β of finding a better model (Eq. 17). The value of β was set to 0.001.

$$i = \frac{\log \beta}{\log(1 - (m/n)^k)} \tag{17}$$

Increasing the number of alignments required in the procedure increases the calibration precision at the cost of increased user strain. As pointed out in [37], it is advisable to track the workload increase using subjective measurements, such as NASA TLX [39]. In our case, we did not consider fatigue a limiting factor during calibration, as we prioritized locational realism. The aim was to obtain a functional procedure which can be used in experimental settings, not to assess the usability implications behind the procedure itself.

After the eye-*mark* projective transformation is found, it is possible to adapt the calibration to any system by finding the ${}^{M}_{O}\mathbf{T}$ transformation (Eq. 18) between the *mark* tracker and the device origin reference frame (see left part of Figure 8). To do this, we temporarily attached a camera (ZED mini) in a known ${}^{M}_{Z}\mathbf{T}$ position with respect to the *mark* tracker with a 3D printed part (Figure 9). We performed a stereo camera calibration with the HMD camera to find the ${}^{Z}_{C}\mathbf{T}$ transformation. The camera can then be removed, as it is contextually needed only to find ${}^{Z}_{C}\mathbf{T}$. The transformation ${}^{C}_{O}\mathbf{T}$ between the HMD camera and the device origin is obtained through the manufacturer documentation.

$$ {}^{M}_{O}\mathbf{T} = {}^{M}_{Z}\mathbf{T}\,{}^{Z}_{C}\mathbf{T}\,{}^{C}_{O}\mathbf{T} \tag{18} $$

Using the *mark*-eye transformation ${}^{M}_{V}\mathbf{T}$ (extrinsic parameters extracted from $\mathbf{G}$), which can be decomposed (Eq.19) into ${}^{M}_{O}\mathbf{T}$ (*mark*-device origin transformation) and ${}^{O}_{V}\mathbf{T}$ (device origin-eye), we can compute ${}^{O}_{V}\mathbf{T}$ to express the eyes position in any device reference frame (see Figure 8).

$$ {}^{M}_{V}\mathbf{T} = {}^{M}_{O}\mathbf{T}{}^{O}_{V}\mathbf{T} \tag{19} $$

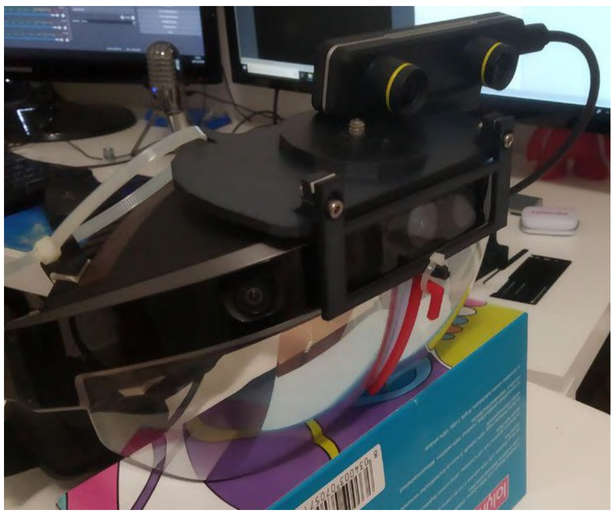


Figure 9: Calibration of the ZED camera with the Meta2 camera to obtain the transformation matrix ${}^{Z}_{C}\mathbf{T}$.

3.0.1.2 *Fine Alignment*

Since the full calibration procedure has been proven to be subject to human error due to the alignment task's difficulty, a general calibration profile has been obtained after minimizing the detected residual error over several calibrations. The OST calibration and validation of the setup have been further discussed in previous studies: for more details, refer to [7, 96]. Each user optical model will still be slightly inaccurate. Thus the residual drift error has to be adjusted by each user by aligning a virtual checkerboard on top of a checkerboard attached to a tracked metal table. The task of aligning a grid, as opposed to aligning several points in a row, decreases the time and fatigue needed for each calibration session, as it is much easier to fit a plane rather than matching a point with pixel-wise accuracy. The residual error correction is performed separately for each eye. The users physically hold the real checkerboard and move/rotate it in different positions, communicating the needed adjustments to the experimenter until a correct alignment is maintained through several roto-translations. A gizmo is displayed to show the virtual checkerboard reference frame's axes and simplify the communication of the adjustments needed. The calibration effects can be seen in Figure 10: the augmented and real objects are aligned.

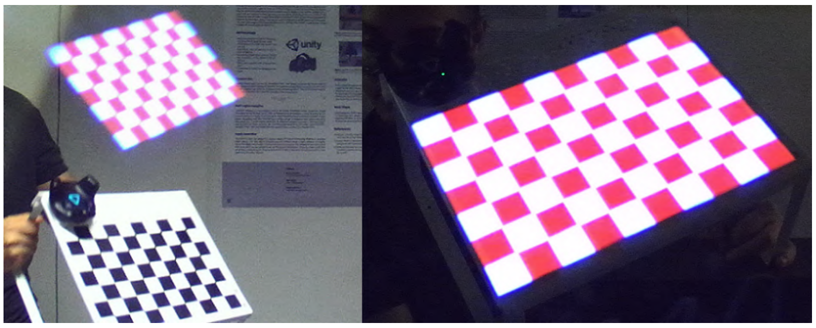


Figure 10: The view from inside the OST HMD before (left) and after (right) the SPAAM and Fine Alignment procedures.

3.0.1.3 *OST Registration Residual Alignment Error*

The Mean Reprojection Error of the stereo camera calibration between the ZED left camera and the OST HMD camera, used to find the ${}^{Z}_{C}\mathbf{T}$ transformation (section 3.0.1/Figure 8) was of 1.72 pixels.

To quantify the OST residual calibration error, we performed several calibrations with a mannequin as described in [7] (see Figure 11a). We then considered (see Figure 12a) the real position of the projected 3D points (blue dots) with respect to the positions of the corresponding virtual points (red dots) of the checkerboard used for the OST calibration (Figure 10).

The average Euclidean distance error (thus, along all 3 axes) between perceived and real positions is 23 ± 11.5 mm, computed over 594 pairs of points (11 image pairs, 54 points per image). When considering only X and Y axes without the depth (i.e., the plane orthogonal to the user's optical axis), the average 2D Euclidean distance error perceived is 8.5 ± 4.5 mm.

The heat map (Figure 12b) of the alignment error distribution along the XY plane (orthogonal to the user's optical axis) shows that the distortion increases towards the boundaries of the field of view. The depth misalignment does not change significantly over distance, with a mean error of 20.5 ± 12.6 mm.

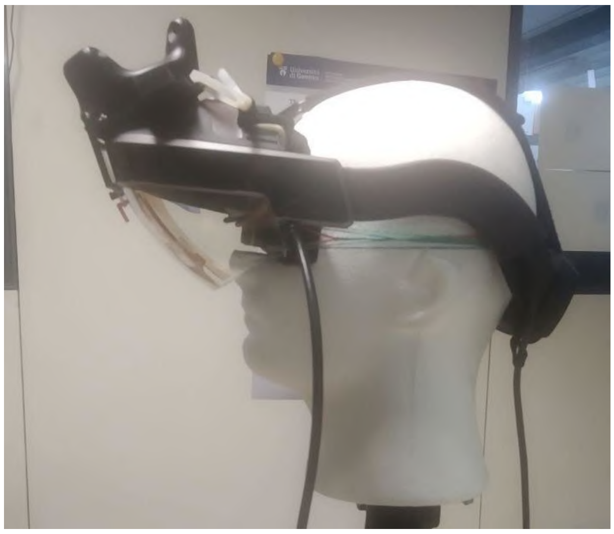
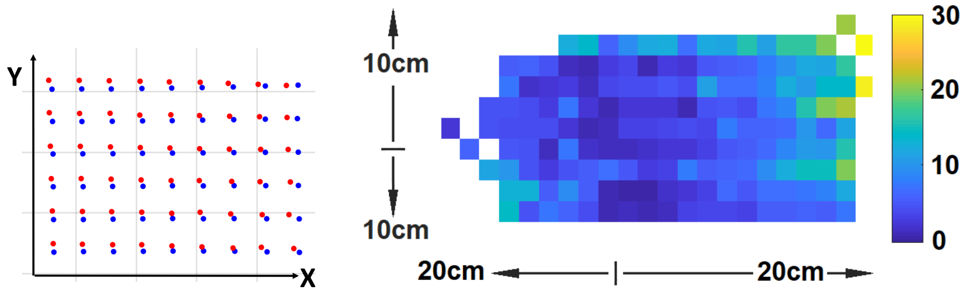


Figure 11: The mannequin head used to validate the HMD registration.



(a) (b)

Figure 12: (a) The points triangulated in the real (blue dots) and virtual (red dots) stereo rigs (only the left view is displayed). The 2D misalignment between the triangulated 3D points can be observed (grid size is 5 cm). (b) Heat map of the Euclidean distance of alignment error along the plane orthogonal to the optical axis (colormap measures in mm).

The OST HMD, registered in the HTC Vive system, runs with a frame rate (computed over 5 minutes with a cumulative moving average) of 74.8 frames per second (fps), comparable to the 76.3 fps of the VST HMD.

3.0.1.4 *OST Registration Module Discussion*

The distribution and magnitude of the misalignment over the image plane (Figure 12) can be used as a metric of the perceived error, which can help define which areas of the workspace are suitable for egocentric interaction. From the heat map representation (Figure 12b), we can observe how the error increases towards the boundaries of the field of view, possibly due to a non-compensated radial distortion introduced by the OST HMD optics. Moreover, during stereo camera calibrations, distortions often manifest towards the image boundaries: since the camera's views lateral boundaries are usually not overlapped due to disparity, checkerboard alignments cannot be performed in those areas, introducing a bias on the reprojection model. Despite a slight constant depth misalignment, we observed that the misalignment was hardly noticeable: we assume that the error is compensated by a scale factor introduced by a focal length drift with respect to the real one. For more details about the OST calibration and validation, refer to [7].

3.0.2 *VST Registration Module*

The VST HMD, composed of the Vive Pro HMD with a mounted ZED-mini stereo camera, uses a non-orthoscopic parallel setup. The optical parallaxes between the user's eyes and the cameras can cause several possible distortions, as described in [90], which can influence the user's perception of spaces.

Since this study aim is not to build an orthoscopic parallax-free VST HMD but rather assess the usability impact of the distortions caused by currently available HMDs, we will not address those aberrations. However, we performed some simple adjustments which should reasonably be performed by every AR HMD utilizer. First, we performed a traditional stereo camera calibration using MATLAB's toolbox based on Zhang's implementation [106], to ensure the ZED video feed was not distorted.

The ZED Mini stereo cameras are then attached on top of the Vive Pro HMD using the manufacturer mount, which encapsulates the Vive frontal cameras with a very tight fit. Our setup is thus the same as a standard user would have. The last adjustment we made is an IPD tune for each user. To do this, the position of each eye's virtual camera was modified in the same way described in the OST calibration, by aligning a virtual checkerboard over a real one (Fig 10) in several poses, one eye at a time. The users could only translate the virtual checkerboard on x and y axes, thus modifying only the virtual stereo camera's extrinsic parameters for the rendering. The VST HMD can be already considered in the proposed registration framework's common reference frame, as the HTC Vive system is already tracking the HMD, and the ZED camera is attached with the manufacturer mount.

3.0.2.1 *VST Registration Module Discussion*

Regarding the VST HMD, only the outer cameras have been calibrated. Thus perceptual aberrations are expected, as discussed in [19, 22]. As previously mentioned, we calibrated the OST HMD only to have a framework where all the obtained data can be coherently compared, but the aim is to assess the standard performance of the two HMDs as interaction devices (as provided by the manufacturers, since most users use them without modifications) and the impact of the optical distortions they introduce on interaction.

3.0.3 *User Tracking: Kinect Color Tracking and Registration Modules*

To detect the user reaching, a Kinect V2 was used. In the unregistered case study [10] discussed in Chapter 4.1, the reaching position was captured through the skeleton tracking features provided by the Kinect sensor SDK, which proved to be subject to several issues. One of them is the sensor's tracking instability, which cannot precisely find the hand position due to occlusions. Moreover, measuring the grasp error is subject to more noise than in a finger reaching task, as it is not trivial to express the perceived position of a 3D point in space by using the palm. Thus, here we detect the position of the finger that is used in the reaching task.

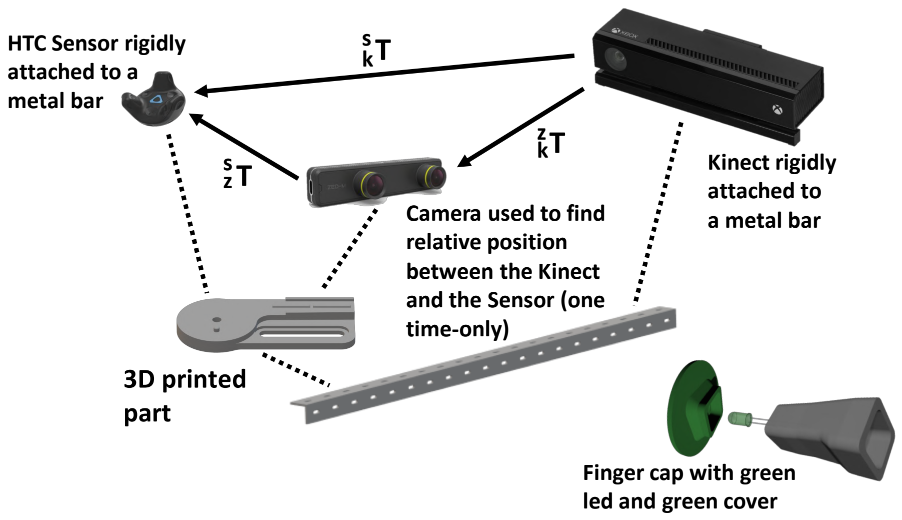

HTC Sensor rigidly attached to a metal bar
${}^{S}_{k}T$
${}^{S}_{Z}T$
${}^{Z}_{k}T$
Kinect rigidly attached to a metal bar
Camera used to find relative position between the Kinect and the Sensor (one time-only)
3D printed part
Finger cap with green led and green cover

Figure 13: A camera (ZED mini) is temporarily placed in a known fixed position ${}^{S}_{Z}\mathbf{T}$ with respect to the Vive sensor, through a custom 3D printed part. The camera is then calibrated with the Kinect camera to obtain the transformation ${}^{Z}_{K}\mathbf{T}$, which can then be used to obtain the transformation ${}^{S}_{K}\mathbf{T}$, needed to convert Kinect coordinates into the HMDs reference frame. The finger-cap tracked by the Kinect contains a green led: a green plastic disk covers the led and slightly increases the finger cross-section to increase the tracking reliability.

To track the user's finger, a color-detection algorithm has been implemented in C++ with OpenCV. A simple slider-based interface allows us to define the HSV color ranges, and the settings are saved for future uses. Once binarized through the HSV filter, the image is processed by a morphological erosion filter to remove noise and a morphological dilation filter to fill small gaps in the tracked finger. Since the experiment was conducted in the darkness, the users were asked to wear a finger-cap with a green led positioned on the fingertip, which facilitated the segmentation. After the filtering, the binarized image will contain only the blob representing the source of light on the finger: the centroid of the blob has been considered as the 3D position of the reaching. Since the Kinect V2 works on infrared light, it can detect distances in the dark without issues. However, it is possible that the image coordinates of the blob centroid are not mapped in the IR sensor since the RGB camera resolution is 1920x1080 pixels while the IR camera has a 512x424 pixel resolution. Whenever the correspondence between the RGB and the IR images was not found, the closest valid pixel was taken instead. To avoid taking ill-measurements due to the small cross-section of the finger during a reaching task, the finger-cap was covered by a small green plastic planar circle, which ensured that the Kinect sensor did not miss the small fingertip surface area (See Figure 13 bottom-right), and also served to filter out the higher light frequencies components of the led light, which were casting several rays in the dark image. Several finger-caps were printed in different sizes to give a tight fit to all the users regardless of the finger size to maintain reaching precision.

The Kinect sensor was positioned 1.5m in front of the users, fixed with a 1/4" screw on a metal bar (Figure 24 left). To convert the Kinect data in the same coordinate system of the HTC lighthouses tracking system, a Vive Tracker was also placed on the metal bar in a fixed position. The Vive Tracker was thus rigidly attached to the Kinect sensor up to an unknown transformation. To find the transformation between the Tracker and the Kinect sensor, the tracker was fixed on a 3D printed part designed to house a camera (ZED Mini). Then, the transformation ${}^{Z}_{K}\mathbf{T}$ between the Kinect sensor and the ZED camera has been obtained through traditional stereo camera calibration. At the same time, the transformation ${}^{S}_{Z}\mathbf{T}$ between the ZED camera and the HTC Vive tracker (S) is known from the CAD of the 3D printed adapter. By chaining the two transformations in cascade (${}^{S}_{K}\mathbf{T} = {}^{S}_{Z}\mathbf{T}\ {}^{Z}_{K}\mathbf{T}$), we can convert 3D coordinates expressed in the Kinect reference frame into coordinates expressed in the HTC reference system, regardless of the position of the Kinect sensor (see Figure 13). The raw 3D measurements obtained by the Kinect were sent to Unity through a UDP connection.

In this way, we can compare the user's finger's position during the reaching task (tracked with the Kinect) with the real position of the stimulus, which is expressed in the HTC reference system, and obtain an error measure.

3.0.3.1 *Kinect Residual Tracking Error*

The Kinect measurement error in the world reference frame (after the transformations) has been obtained by positioning the rod with the light box in 27 distinct positions, which represent the possible positions that have to be reached by the user's finger in the experimental setting described in Chapter 4.2. The positions evenly span over the interaction volume, to obtain an error measure which directly refers to the experiment performed. Refer to Chapter 4.2 for further details on the experimental setup.

First, we measured the ground truth coordinates by attaching a Vive tracker on top of the light. Then, we removed the Vive tracker and attached the finger-cap with the green LED right on top of the correct position, touching the light box. Finally, the box was removed, and the Kinect detection of the finger was captured. The 27 measurements in world coordinates obtained by the Kinect were then compared to the 27 measurements taken with the Vive Tracker.

Calibrated Cameras	MRE
ZED left and right cameras: to calibrate the ZED stereo rig (section 3.0.1/3.0.2)	0.59
ZED left camera - Kinect camera: to find the ${}^{Z}_{K}\mathbf{T}$ transformation (section 3.0.3/Figure 13)	0.35

Table 1: Mean Reprojection Error (MRE) of the stereo camera calibrations used to register the Kinect, representing the distance (in pixels) between a pattern key-point detected in a calibration image, and a corresponding world point projected into the same image.

The mean Euclidean error between the measurements was 15 ± 12 mm, which we treat as the accuracy error. To assess the precision of measurements, we collected another 3 batches of measurements of the 27 grid positions with the Kinect. We computed each point's mean position (separating the three axes) and the absolute difference of the 4 readings with respect to the mean value. We then computed the mean absolute difference of all 27 points with respect to their mean position and obtained a precision error of 1.7 ± 2 mm on the x-axis, 3.4 ± 3 mm on the y-axis, and 4.4 ± 4 mm on the z-axis.

Finally, the mean reprojection errors (MREs) of the stereo camera calibrations are reported in Table 1.

3.0.4 *Real Case Registration Module*

During interaction tasks, the reaching misalignment error cannot be imputed solely to the reduction of optical cues induced by the use of an HMD, as the reaching also involves a motor task. As we already stated, the control group's motivation is to see the impact of any biomechanical bias during the movement and isolate the real perceptual error induced by the HMDs. The control group had to perform the same reaching task but without HMD.

To be as coherent as possible to the AR case, we devised a system to provide the same visual stimulus streamed through the HMDs (Figure 15).

The visual stimuli were provided using a LED light connected to an Arduino Uno and programmed to turn on for the same amount of time as in the AR task (1 second). A remote controller controlled the LED light, and the whole circuit was encased in a 3D-printed box designed to be precisely positioned on 3 specific positions of a metal rod. The box had a circular hole (1 cm radius), which was internally covered with a small square of plastic (0.8 mm thick) to diffuse the LED light in a uniform circle, to be as similar as possible to the stimuli provided in AR. The metal rod was then placed in fixed positions of a metal structure with 18 hook supports and served as support for the rod. Thus, the LED light could assume 3 different positions on the rod, which could be housed in 9 different positions, achieving the same $3 \times 3 \times 3$ stimuli grid that was displayed through AR HMDs (see Figs. 14 and 24). The experiment

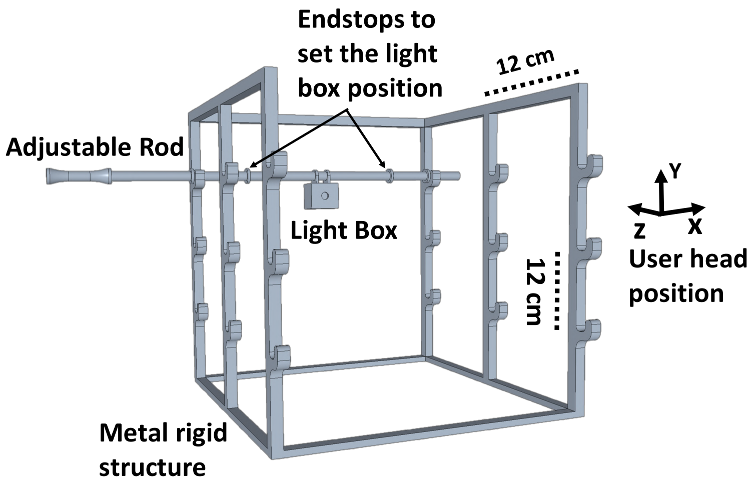


Figure 14: The structure used to convey visual stimuli similar to the ones perceived in AR, while not wearing any HMD (see Figure 15 for a depiction of the perceived stimulus). A box with a green LED can slide on a rod and snap into 3 possible positions. The experimenter can move the rod into 9 possible positions, thus achieving a $3 \times 3 \times 3$ grid, enclosed inside a cube with a side length of 24 cm.

was specifically conducted in the darkness as the whole metal structure's sight would otherwise provide a strong cue on the stimulus's 3D position.

The positions of the grid points, expressed in the HTC Vive reference system, have been obtained by positioning the LED box in all the 27 positions and measuring its 3D world coordinates with a Vive Tracker attached on top of the center of the LED light. The whole structure was designed to quickly remove the metal rod after the visual stimulus vanished, leaving an empty volume in front of the user. In this way, we deprived the users of the haptic feedback deriving from the finger's physical contact with the LED box.

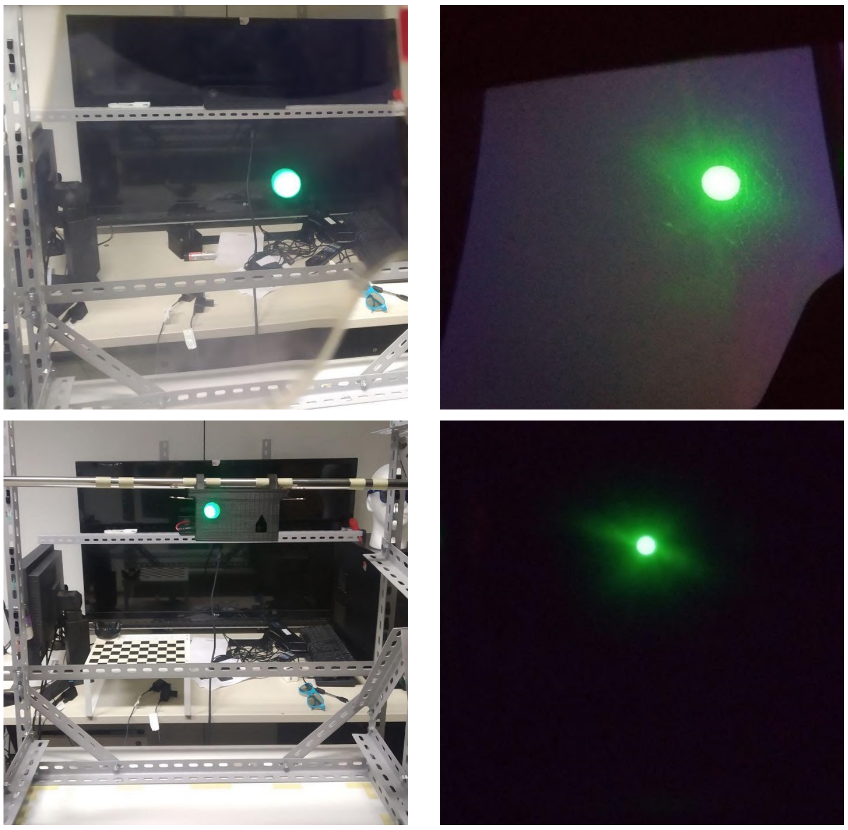


Figure 15: Top: the stimulus as observed from inside the OST HMD through an inner camera. Bottom: the stimulus as observed without any HMD, displayed through a green LED, using the structure displayed in Figure 14. On the right side, the views in the darkness are displayed, as during the experiment. In the darkness, the perceived stimuli are the same.

4 Perception and Action in Peripersonal Space

The previous chapter describes the design and development of a registration framework which can be used to develop AR applications in such a way that the user, the environment and the virtual content are coherently co-localized and synchronized. In this chapter, we describe how we used the developed framework to carry out further research on the perception/interaction loop in peripersonal space.

The next three sections (4.1, 4.2 and 4.3) thus describe the studies specifically designed, implemented and carried out to understand how people perceive and process information in AR environments and which is the best interface in terms of transparency, able to guarantee a minimum cognitive load on the user so that he can concentrate on the current task.

Three cases are considered: the first one (section 4.1) is a reaching task performed with unregistered devices. The aim is to quantify whether distortions in perception of the spatial layout of the scene occur, by taking into consideration two different AR HMD (VST and OST). The considered devices are Samsung Galaxy S6 with a DivineViewer VR headset (Google Cardboard) used as VST HMD and a Meta2 as OST HMD.

The next one (section 4.2) is a blind reaching task, carried out in a more controlled environment with the proposed registration framework described in chapter 3. This experiment aims to isolate the visual perceptual error in the reaching task by removing the visual alignment cue by using the blind paradigm. The experiment is performed in the darkness to be able to have a coherent control group, who do not use any HMD, to have a baseline with which to compare the performance of the users when wearing HMDs. This time, both HMDs are binocular and within the same price range, to have a better comparison between currently available AR HMDs. The used devices are an HTC Vive pro with a ZED mini frontal stereo camera as VST HMD and a Meta2 as OST HMD.

The last one (section 4.3), also performed with the proposed framework, is an interactive task. This experiment aims at obtaining a comparison of the two AR devices in a functional and interactive task. The aim is to assess whether and how the distortions observed in the blind reaching experiment affect user interaction in more naturalistic settings, where multiple depth cues are available. The used devices are the same seen in section 4.2, as the aim is to compare the performances of the two HMDs in different settings.

4.1 Using Stock, Unregistered AR Devices

In this experiment, participants were standing (to not give any haptic or perspective clue) and tried to reach some virtual cubes generated randomly within a fixed 50x30x30 cm volume, being as precise as possible (see Figure 16). The user had 5 seconds to adjust the position for each tentative. To a subset of the participants, the visual feedback about how the hand was registered internally was not provided (occlusion handling, i.e. hand augmentation, was disabled), thus they had to rely only on their perception of virtual objects and own real hand (Test I - *without feedback*).

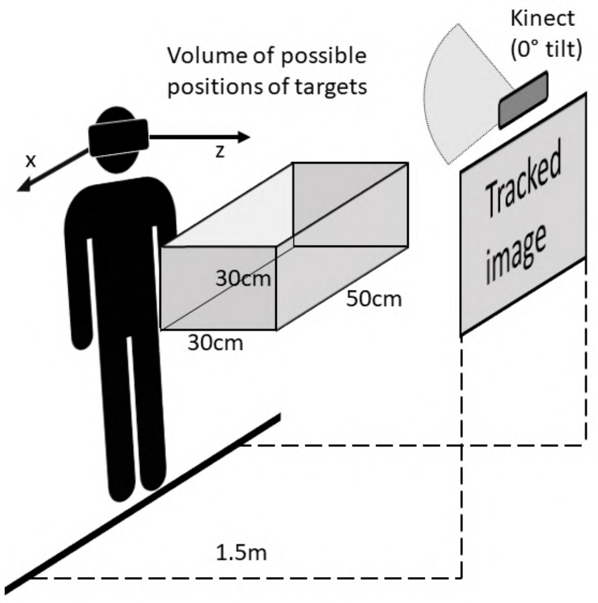


Figure 16: Scene sketch of the experimental setup.

To the remaining subset of participants we gave a feedback of the user's hand position in the internal reference frame (Test II - *with feedback*). In both systems the participants could see when the hand was entering inside the cube. With the OST, the depth occlusion handling (which relies on the embedded depth sensor) was enabled again. With the VST, a red boxing glove was rendered in such a way that it approximately overlapped with the real hand (Figure 17).

Each participant was also asked to compile a Simulator Sickness Questionnaire (SSQ, [52]) before and after using each device.

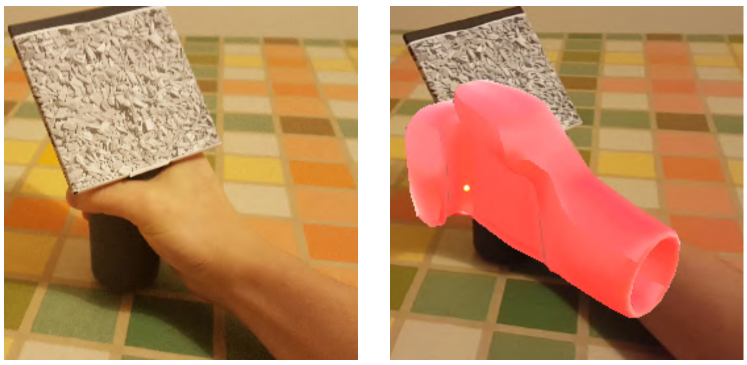


Figure 17: The virtual augmentation of the hand for the VST device. The users have been asked to grasp the handle in such a way that they perceived the glove as overlapped to the real hand.

4.1.1 *Setup*

A Kinect device was rigidly attached to an horizontal frame with 0° tilt (pointing straight forward) at a 156.5 cm height (measured from the ground to the sensor center). The Kinect was facing the subject, 1.5m away from his point of view: participants had to stand on top of a line taped on the ground, which signaled the 1.5m mark (Figure 16). To reduce the amount of noise in the data obtained by the Kinect sensor, the movements of the subjects were restrained. Participants were standing in a precise location in a small room and were asked not to move their feet.

4.1.1.1 *Video see-through*

The considered HMD is a Samsung Galaxy S6 with a DivineViewer VR headset (Google Cardboard). The rear 16MP camera was used to capture the video feed. The S6 screen resolution is 2560x1440 pixels (1280x1440 per eye) with 30Hz refresh rate. The field of view of the HMD is $\sim 95°$. The AR tracking was performed using Vuforia integrated in Unity. Vuforia is an AR SDK which uses computer vision technology to recognize and track planar images and 3D objects in real time. It is then possible to position and orient virtual objects with respect to the reference frame attached to the tracked real world objects, when they are viewed through the camera of a mobile device. In our case, two images (targets) were tracked simultaneously: the first one was a 420x594mm picture attached vertically, just below the Kinect sensor. The center of this image served as global reference frame for the inner game measures. The second image target was sized 90x100mm, and it was needed to track the hand position. This image was attached to a 3D printed handle, which was a 10x10 cm plane attached to a 10 cm tall cylinder with 45° slope. The handle was designed to be easily seen by the camera from a seated position, to make the tracking more reliable. Vuforia does not currently handle occlusions between real and virtual objects: the augmentation is always overlapped on top of the video feed, regardless of their true relative positions. Thus, the augmentation of the hand was helping the user by providing back this depth cue, since the relative occlusion between virtual objects can take place.

4.1.1.2 *Optical see-through*

The used device is a Meta 2 headset by MetaVision. The Meta HMD has 90° field of view and a resolution of 2560x1440 pixels (1280x1440 per eye), similar to the VST device. In contrast to the VST device, however, the OST FOV only refers to the area where the virtual augmentation can take place; the user can still see through the remaining portion of the headset's glass. The front camera, used for the inner SLAM tracking, has a 1280x720 pixel resolution. MetaVision's HMD is designed to work best within arm's reach (eyes vergence is not dynamically tracked).

4.1.1.3 *Experiment overview*

The scene was similar for both HMDs, with a few differences. A cube was rendered in a random position inside a volume of 50x30x30 cm size, at discrete intervals of 10 cm. Thus each cube had 72 possible positions. The spawning volume position was slightly shifted in position between the two HMDs due to internal differences of how the two devices' tracking works (e.g. VST HMD has problems to detect the image on the handle close up simultaneously to the bigger image 1.5m away, as the smartphone can only keep one of them in focus). Each cube was 10x10x10 cm wide for both HMDs, and disappeared after 5 seconds of continuous detection of the user's hand. Upon destruction of a cube, its position and the hand current position, which was given by the hand tracking in the case of the OST and by the handle target tracking for the VST, were recorded internally into a text file. This data is thus referred as *internal data*, as these measures are entirely dependent on the tracking performed by the Vuforia and Meta SDK. After a cube disappeared, a new one instantly appeared, in a different random position.

The scene is also acquired by an external device, i.e. a Microsoft Kinect, in order to obtain quantitative measurements not dependent on the VST and OST systems. Once acquired, we segmented the Kinect data by first derivating the measures over the height. Such derivative is close to zero only when the subjects held their hand position still inside the cube, or when they had lowered it to the side, waiting before trying the next grasp. The cubes were appearing instantly one after another, but sometimes subjects had to look around as they could appear slightly outside the FOV. Hence we extracted the segments of data that satisfied two conditions: (i) the derivative over the height was close to zero; (ii) the height of the hand was above the average height of the hand when resting on the side, which was recorded before starting the test for each subject.

Each extracted position was thus composed of all the readings, starting from the first time the above two conditions were verified, until one of the conditions ceased to be valid. Sequences of data shorter than 50 readings were discarded to remove outliers: since the subjects maintained the position for about 5 seconds and the Kinect samples at 30hz, valid sequences had to have around 100-150 readings. The final extracted positions have been computed from averaging the readings of each segment. We computed the average depth of the targets that appeared for both groups, as well as the average depth obtained both from the data gathered internally and the data registered with the Kinect sensor.

Finally, the SSQ [52] has been analyzed by computing the four indexes of nausea, oculomotor, disorientation and total cybersickness score.

4.1.2 *Procedure*

A total of 45 volunteers participated in the experiment.

The first 30 volunteers (10 females) performed the Test I (without feedback), over two days. The VST was tested during the first day by 15 participants (5 females). They were aged between 19-29 years (mean 23.0 ± SD 2.8 years) and between 161-188 cm tall (mean 173.8 ± 8.0 cm). The OST was tested during the second day by 15 participants (5 females). They were aged between 19-29 years (mean 24.7 ± SD 3.0 years) and between 156-188 cm tall (mean 173.9 ± 8.7 cm). The subjects were not tested for eyesight, and everyone wore the headsets as they were most comfortable.

The remaining 15 volunteers (5 females) performed the Test II (with feedback), and were tested within a day. This time, all of them tried both devices. Each participant started with a different HMD than the previous one (to not introduce bias due to eye strain or learning processes).

The IPD of subjects was not measured: an average value of 63mm was used to render the two views. The HMDs were large enough that all the subjects who had glasses managed to wear them underneath the device.

The experiment has been conducted inside our University laboratory: participants were mostly ranging from undergraduate to PhD students.

The subjects, after giving informed consent, first compiled the SSQ with their initial conditions before using the HMDs. After noting age, height and gender, each candidate was told to stay still on top of the line which marked a 1.5m distance from the Kinect sensor and Vuforia image target. The subject standing position was measured for a few seconds by the Kinect before starting the test. A brief explanation was given about how to use and wear the HMD, and the structure of the test: the participant had to move the hand inside the perceived position of the cube, stand still for 5 seconds, and lower the hand to the side before trying again with the next one. Each participant was given 5 minutes, thus made approximately 50-60 reaching attempts at most. The VST device was sometimes refocusing the camera resulting in a little slower tracking, thus participants who used the VST HMD ended up making a little less reaching attempts than those using the OST HMD (approximately 40-50). After finishing the test, the subject was asked to compile again the SSQ.

4.1.3 *Data Analysis*

Figure 18(c,f) shows the distribution of hits centered around the goal: VST has a higher spread with respect to OST. We computed a linear fit of the hand positions with respect to the real positions of the cubes (on the z axis) for each participant (see Figure 20(a,b)), and plotted a normal distribution of slopes by using the average and standard deviation values for each group (see Figure 22), similarly to a previous study that used the same approach [65].

4.1.3.1 *Test I - without feedback*

By looking at the plotted positions of the hands registered both internally and by the Kinect for the OST HMD, we can notice no particular difference between the two measures, and a slight depth compression (see Figure 18(d,e)). The mean depth errors (Tab.2) confirm the presence of an underestimation of a few (∼6) centimeters. There is a trend towards a larger underestimation as the distance from the user increases (Figure 20(a,b) and Figure 21(e,f)). On the other hand, there is a significant depth compression with the VST HMD. Moreover, the internal data seems to be slightly mismatched with respect to the Kinect data for the measured depths (Figure 18(a,b); Tab.2). This suggests that there could be a misalignment between the internal scene view and the real world.

A two sample t-test revealed significant difference ($p < 10^{-7}$) between the average slope of the two HMDs: the mean slope for the VST HMD was $16.57^\circ \pm$SD 9.13°, while for the OST $37.46^\circ \pm$SD 4.66°. It is worth noting that 10 out of 15 subjects who used the VST HMD obtained slopes under 20°, suggesting consistent inability to estimate precisely the depth of virtual objects if no other internal feedback is given, and lack of perceived difference between two objects at different distances. All the subjects who used the OST HMD obtained slopes above 25°. There were no slopes over 45°, nor negative ones, thus on average none of the subjects overestimated depths.

4.1.3.2 *Test II - with feedback*

The distribution of hits centered around the target for VST HMD users was more condensed with respect to Test I (Figure 19(c) and Figure 18(c)), which was to be expected. The comparison between the same distributions for OST HMD users reveals no appreciable difference (Figure 19(f) and Figure 18(f)).

The slopes obtained by the single subjects also show an improvement for VST HMD users and a consistency with previous results from OST HMD users. From those who used the VST HMD, 5 out of 15 obtained slopes under 20°, as opposed to 10 over 15 of Test I. Only one subject that used the OST HMD obtained a slope under 20°.

The SSQ analysis reveals a slight increase in cybersickness symptoms when using the VST HMD, and no significant change when using the OST HMD (see Figure 23 (a,b)).

Test	HMD	Data source	MAE of perceived depth
I	VST	Internal	19.2±10.2 cm
I	VST	Kinect	28.4±15.2 cm
I	OST	Internal	5.8±10.1 cm
I	OST	Kinect	5.8±11.2 cm
II	VST	Internal	5.4±10.3 cm
II	VST	Kinect	17.4±14.8 cm
II	OST	Internal	3.0±9.6 cm
II	OST	Kinect	0.4±10.3 cm

Table 2: Comparison of VST and OST depth Mean Absolute Error (MAE).

4.1.4 *Outcome*

We analyzed the perceptual difference between VST and OST HMDs users, gathering data from 45 volunteers. We subdivided the analysis into two different tests.

In Test I - *without feedback*, we positioned the reaching goals in front of the subjects, thus the perspective cue cannot help, but only the perspective size. Here, we measure the reaching positions by using a Kinect device in order to have the positions with respect to an external reference frame. We assess the possible misperception of the spatial structure of the scene: for this test we do not use augmentation of the subject's hand, thus we allow only a visual online error correction between the real hand and the AR contents. This can be useful to understand whether it is possible a natural interaction between real and virtual objects. VST shows worse performances and also OST decreases its reaching precision (see Figure 20(a,b) and Figure 21(e,f)): this can be due to lack of perspective cue and to the frontal reaching task. By considering external measures, VST has larger error in depth with respect to OST (see Figure 18(b,e)): in particular, VST has a mean depth error of ∼30 cm, and OST ∼6 cm (see Table 2). This result suggest that OST allows an effective interaction in AR.

In Test II - *with feedback*, we introduce an augmentation of the subject's hand, thus a visual online error correction between the augmented hand and the AR contents is possible in the same reference frame. In this way, the internal errors decrease for VST and are similar to the previous ones for OST, and the external measures show an improvement of the reaching for both VST and OST (see Figure 19). Though, the error in depth of VST is still large, ∼17 cm. Whereas, OST is ∼1 cm (see Table 2). It is worth to note that since a visual online error correction is possible, the errors are mainly due to the 3D reconstruction of the real environment performed by VST and OST, i.e. how they are able to place virtual objects in correct positions with respect to the real environment.

By looking at the average slopes obtained during the experiment (Figure 22), we can notice how participants who used an OST HMD obtained an overall better result (i.e. the slope of the linear fitting between real measurements and perceived ones is around 45 deg as expected). A two sample t-test revealed a significant difference between the average slopes obtained when using the VST HMD in the conditions described in Test I and Test II ($p < 0.05$). The mean slope in the former case was $16.57^\circ \pm$SD 9.13°, while in the latter it was $26.38^\circ \pm$SD 10.71°. There is no significant difference in perceived depth between users who used the OST HMD in Test I (mean slope $37.46^\circ \pm$SD 4.66°) and those who did it in Test II (mean slope $36.09^\circ \pm$SD 6.64°), suggesting that the OST HMD gives enough depth cues even without the hand augmentation. Moreover, a t-test revealed significant difference ($p < 0.05$) between the average slopes when using the two different HMDs of Test II, which considering previous results, suggests a general better depth perception by subjects who used an OST HMD.

Furthermore, by looking at the mean depth errors registered internally with respect to the external Kinect measures (Tab.2), we can notice how the VST HMD internal measures have an error $\sim$18 cm. Thus, even if the user is able to minimize his error when interacting with virtual objects thanks to an internal feedback, such mismatch with the real measures could make the VST HMD less suitable for haptic applications (e.g. virtually augmented real tools). The OST HMD did not display such behavior, as the mean depth errors measured internally and externally are consistent with each other.

The SSQ analysis reveals a slight increase in cybersickness symptoms (especially oculomotor and disorientation factors) for VST HMD users (Figure 23(c)), although to a lesser extent than the Test I (Figure 23(a)). OST HMD users, on the other hand, did not experience any significant condition aggravation on both tests (Figures 23(b,d)). Thus, we can conclude that the OST HMD is generally better in terms of eye strain and fatigue.

By considering the previous discussions, the results suggest that the considered VST setup does not allow an effective interaction in AR, since there is a shrink of the perceived space, specifically a large underestimation of depth. This fact hampers even the reaching of the AR contents. However, Test II allows us to infer that it is possible an interaction with AR contents through the augmentation of the subject's hand, since this provides a cue for a visual online error correction. Conversely, the results obtained by using OST suggest that such a technique allows effective interactions with AR contents with or without the hand augmentation, thus OST can be considered for interaction also with augmented real objects. A further extensive evaluation, by considering binocular VST devices, may give us further experimental evidence for the comparison of distance perception and interaction with HMD AR devices.

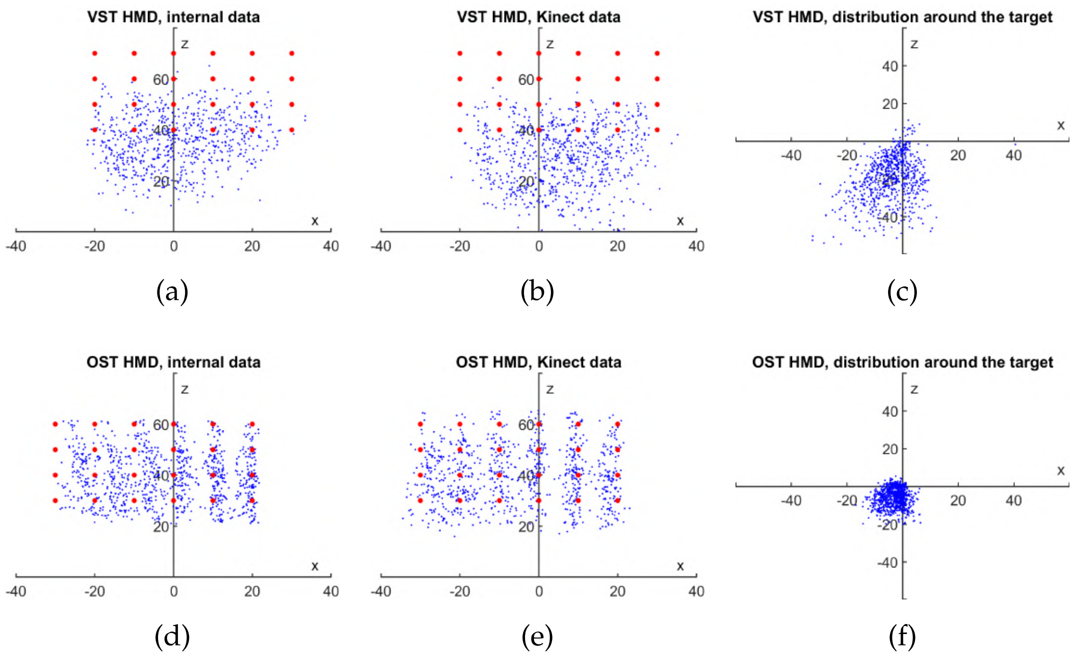


Figure 18: Data from Test I - *without feedback*. (a) and (d) represent the distribution of subjects' hands positions with respect to their standing position in (0,0) - red dots represent the spawn positions of the targets. (b) and (e) represent the same distribution, but with the average hand positions measured by the Kinect. (c) and (f) are the distributions of the hand positions (from the internal data) around the target, which is centered in (0,0), after holding the hand position for five seconds.

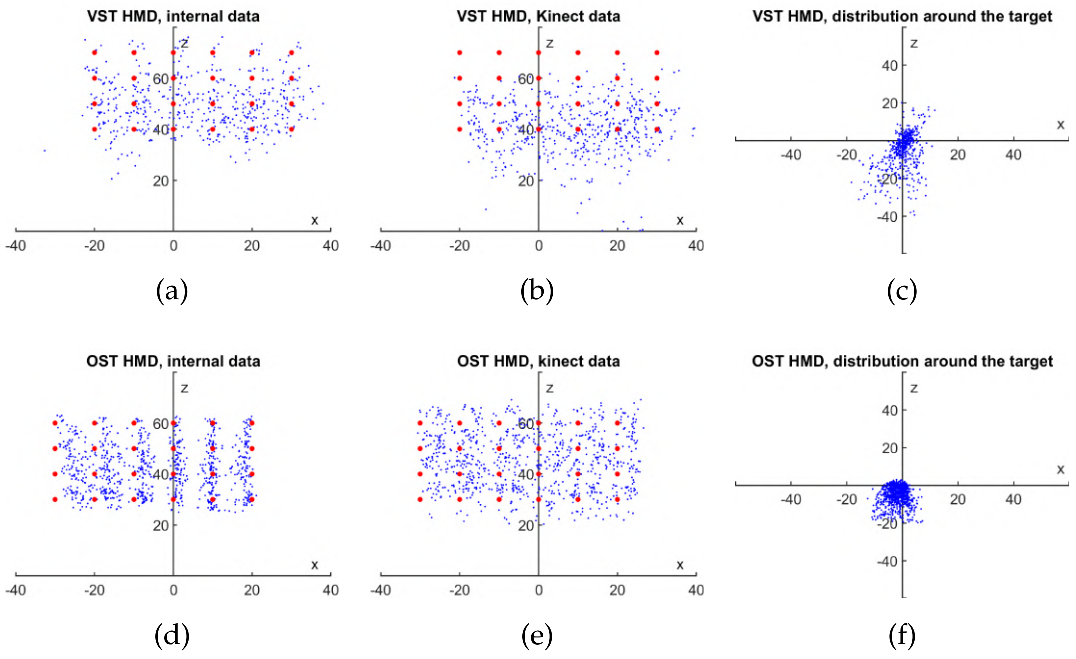
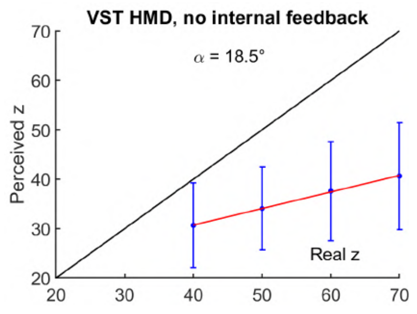
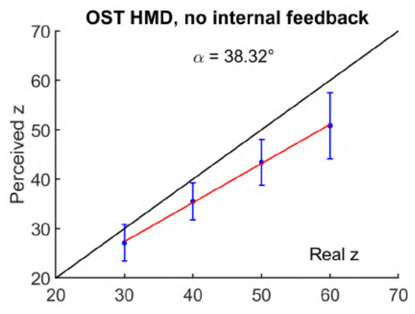


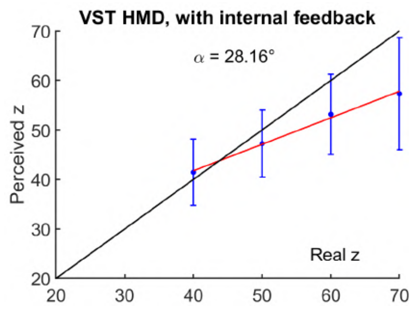
Figure 19: Data from Test II - *with feedback*, displayed in the same order as Figure 18.



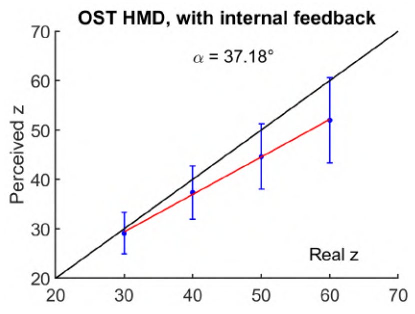
(a)



(b)



(c)



(d)

Figure 20: Difference between real and perceived depth distances (i.e. perceived z distances plotted against the corresponding real ones) in Test I - *without feedback* (top row) and Test II - *with feedback* (bottom row). Left images are from the VST HMD, right images from the OST HMD. The reference α value should be 45 deg, indicating a perceived depth equals to the real one. Smaller α values indicate a compression of the perceived distances.

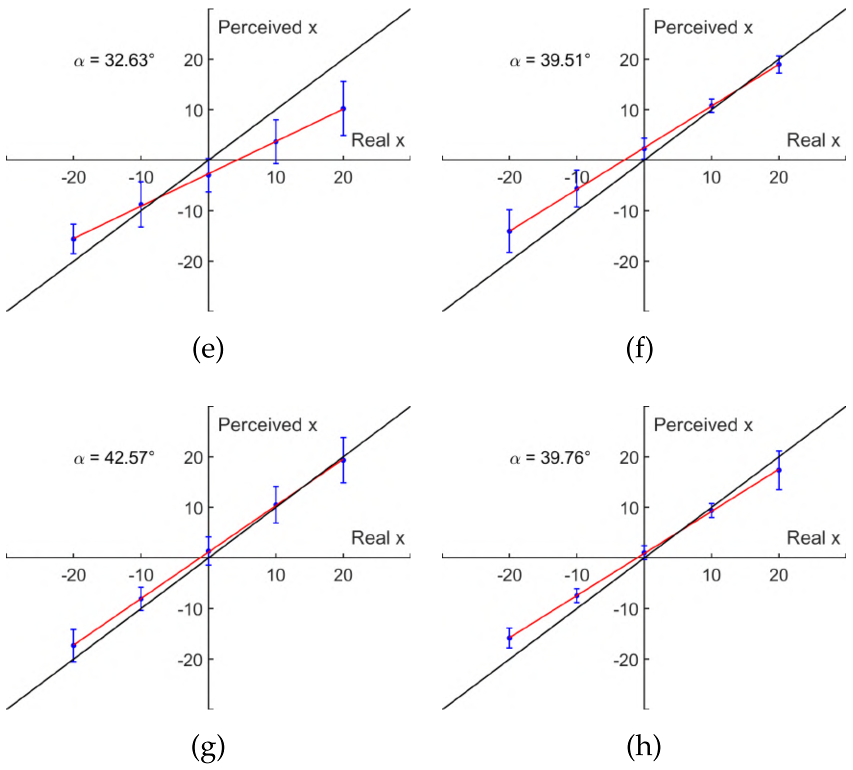


Figure 21: Difference between real and perceived distances on the x axis (left and right with respect to the user) in Test I - *without feedback* (top row) and Test II - *with feedback* (bottom row). Left images are from the VST HMD, right images from the OST HMD. The computed α is reported with respect to an ideal value of 45 deg, and smaller α values indicate a compression of the perceived distances, as in Figure 20.

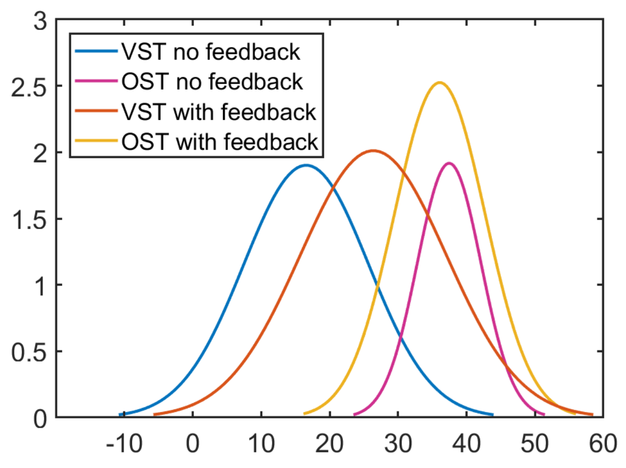
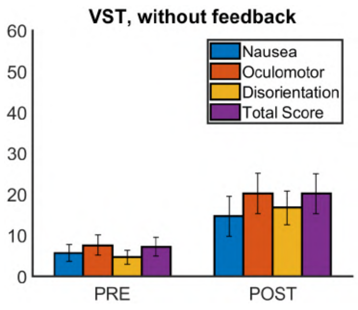
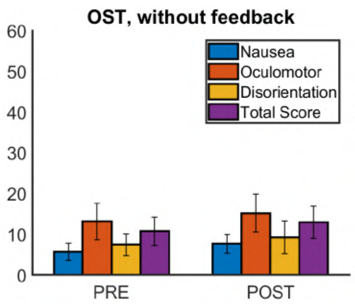


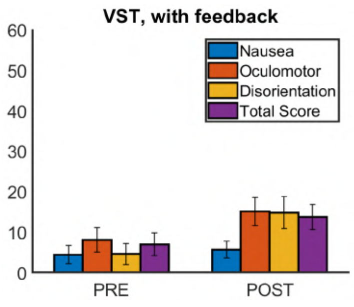
Figure 22: Distributions of slopes (in degrees) between the participants, when using different HMDs and different experimental conditions. Participants who used an OST HMD obtained an overall better result (i.e. the slope of the linear fitting between real measurements and perceived ones is around 45 deg as expected).



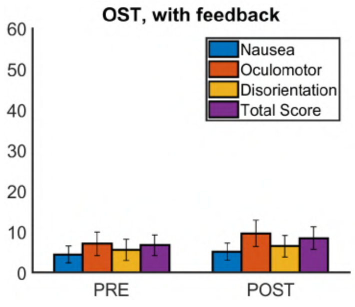
(a)



(b)



(c)



(d)

Figure 23: SSQ results for Test I - *without feedback* (a,b) and Test II - *with feedback* (c,d). PRE and POST refer to answers before and after the experiment, respectively.

4.2 Using Registered Devices without Feedback

This experiment aims to isolate the visual perceptual error in the reaching task by removing the visual alignment cue by using the blind paradigm. The experiment is performed in the darkness in order to be able to have a coherent control group, who do not use any HMD, to have a baseline with which to compare the performance of the users when wearing HMDs. Displaying a 3D point in space without an AR HMD always requires the presence of a structure or contraption, which would otherwise introduce a perception bias if not hidden from the user. Although this experiment can be considered as not strictly AR, as the real-virtual blending is basically disabled, we consider this setup required to establish the impact of perception or other biases present in our experimental setup during the interaction without the visual alignment cue between the target and the finger. We thus have to assess if, in our experimental settings users without HMDs are able to accurately point a target whose location is specified by visual information alone (as in [93]), and blind reaching has been repeatedly found to be more accurate than explicit judgements (e.g. [102]). If distortion is also observed in the control group, then part of the misperception can be imputed to the specific experimental setting, e.g., due to physical constraints during the reaching, such as targets being too far away. Otherwise, any other statistically relevant misperception encountered can be attributed solely to the optical aberrations caused by the two HMDs.

4.2.1 *Setup*

The subjects had to perform a blind reaching task: they were asked to reach with a finger the 3D location of a sphere (1 cm radius), which was shown for 1 second. The experiment is performed with the VST HMD, with the OST HMD, and with no HMD. The experiment was performed in an enclosed room, which was darkened as much as possible by covering any source of light. Since the eyes quickly adapt to dim light conditions, we asked them to keep the eyes closed to deprive any geometrical cue subjects. The subjects were informed when to open the eyes to see the stimulus's position and had to close the eyes again before attempting the reaching (blind reaching task). The same procedure has been used for both the HMDs and the baseline (without HMD), to maintain the same cognitive load over the different tasks and avoid any possible bias.

4.2.2 *Procedure*

We gathered data from 15 volunteers, mean age of 29 ± 7 years (6 females, 9 males). The 15 subjects were evenly split into three groups to avoid bias, and each group tested all three cases in a different order (balanced within-subject experimental design). The average height was 174 ± 10 cm. All the participants had a strong technical background, which allowed the understanding of all the experimentation phases under strict conditions to reduce the noise caused by ill-calibrations. All the subjects had normal or corrected to normal eyesight. Users with glasses were asked to keep them underneath HMDs.

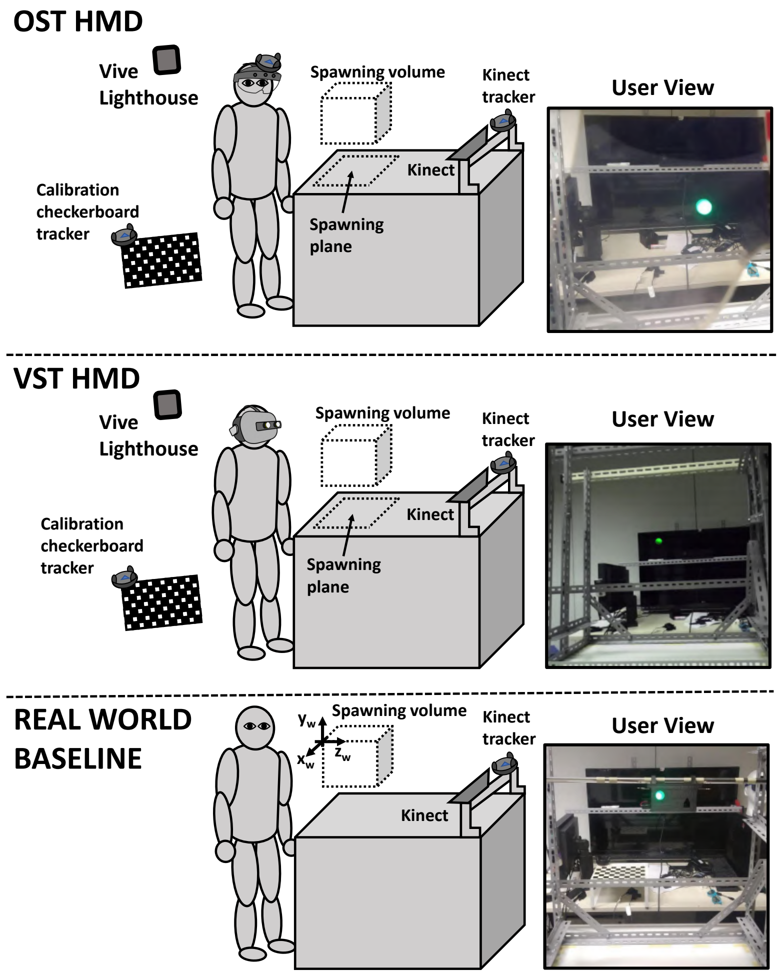


Figure 24: Sketches of the experiments. (left) in the blind reaching task configuration, targets (green circles) appear inside a spawning volume in front of the user: the fingertip is tracked by a Kinect placed in a tracked position. In the interaction kitchen task configuration, targets (kitchen objects, see Figures 27 and 28) appear on the spawning plane; in this configuration, the observer's line of sight is approximately perpendicular to the spawning plane. The HMDs calibrations are adjusted by aligning a tracked checkerboard. (right) an example of the stimuli displayed in the blind reaching task; during the experiment, the room was darkened (as in Figure 15, right images)

The experiment was performed as follows. The whole experimental procedure is explained to the subject before starting. The user is shown how to wear and tighten the HMD (if any is used). Once comfortable, the user is asked to sit still, and the procedure starts by registering the head height. The user is then asked to perform the calibration as described in Section 3.0.1 and Section 3.0.2. Once finished, the subject wears the finger-cap on the right-hand index finger, and the battery-powered LED inside the finger-cap is turned on, and the room lighting is switched off. The subject is asked to close the eyes. The experimenter warns the subject when to open the eyes, just as a target (a 1 cm radius sphere) is about to appear in a random position of a $3 \times 3 \times 3$ grid (of size $24 \times 24 \times 24$ cm). The grid occupies a cubic spawning volume (Figure 24) volume standing 48 cm away from the user. The volume height adapts to the height of the user, which was measured when the experiment began. The target positions are extracted from the randomly shuffled list of the 27 possible configurations, and each target appears only once. The target disappears after 1 second: the user is asked not to move until the target has disappeared and to close the eyes again as soon as the target disappears. Thus, the user blindly reaches the perceived position, and the experimenter records through a button press the finger location and the target position. Each user performed 27 reaching movements per session, and 4 sessions were repeated consecutively, using the same hardware configuration for all 4 sessions. Once the experiment ends, the subject is given enough time to accommodate and rest before testing the other devices configurations (1 day) to exclude any bias caused by the user's fatigue.

4.2.3 *Data Analysis*

The users' fingertip 3D positions during the reaching task have been compared with the real positions, where the stimulus is displayed. Each one of the 15 users performed 27x4 reaching attempts for all the 3 experimental conditions: OST, VST, and real-world (RW) baseline (i.e. performing the task without HMD, our control condition). A total of 1620 points were displayed for each setup. After discarding outliers (mainly due to erroneous recording of the finger position), the final analysis has been performed on 1399 points for the control condition, 1472 points for the VST HMD, and 1361 points for the OST HMD.

Figure 25 (left images) shows the scatter plots of the reaching points (colored dots) with respect to the position of the target stimulus (red dots) for the XY plane (which is a frontal plane for the observer). Figure 25 (right images) and Figure 26 display the horizontal plane XZ's behavior, where the depth is along the z-axis (see Figs. 14 and 24).

Table 3 shows the mean absolute errors (MAEs) during the reaching task for the three axes. With respect to the user, X is the lateral axis (left/right), Y the vertical axis, and Z the longitudinal axis (depth).

Being the data not normally distributed, we performed a Wilcoxon signed-rank test to analyze the obtained errors. The test has shown no significant difference in the y-axis MAE between the OST and Real-World case, and significant difference in all other condition comparisons ($p < 0.005$): when comparing the errors along the axes between the control group and the two HMD, and when comparing the two HMD together.

	OST HMD	VST HMD	Real World
X	33 ± 25 mm	44 ± 29 mm	24 ± 21 mm
Y	31 ± 28 mm	35 ± 29 mm	29 ± 23 mm
Z	98 ± 67 mm	69 ± 52 mm	40 ± 29 mm

Table 3: Mean absolute errors of the Blind Reaching Task, with respect to each axis. Axes X and Y represent the plane orthogonal to the user's optical axis, with Y pointing upwards and X to the right, while the Z axis represents depth, pointing away from the user (see Figure 24).

4.2.4 *Outcome*

In Figure 25 we can observe how the OST and real-world (RW) conditions are comparable as cloud shape for the frontal plane XY, and the error is quite uniform around the target. In contrast, the VST shows a slight asymmetry that is highlighted by the red arrows. This could be due to the ZED camera's not optimal positioning on the Vive HMD, which introduces a rotational error that the user's calibration has not completely removed.

By looking at Figure 26, which considers depth (z-axis), we can notice a different behaviour for the three conditions: indeed, for the OST, the users experience a relevant depth compression. This is evident in the scatter plot (Figure 25, OST top view) and in Figure 26 OST HMD, where the underestimation of depth changes as a function of distance. On the contrary, the VST HMD, while still performing significantly worse than the real world condition, seems to cause less distance compression, and such compression is less affected by the distance. As expected, there is no asymmetry and compression for the RW condition.

The VST HMD data seems to show the effect of the aforementioned rotational error, with a trend to cause an overestimation of the distances over the x-axis (Figure 25, middle plots).

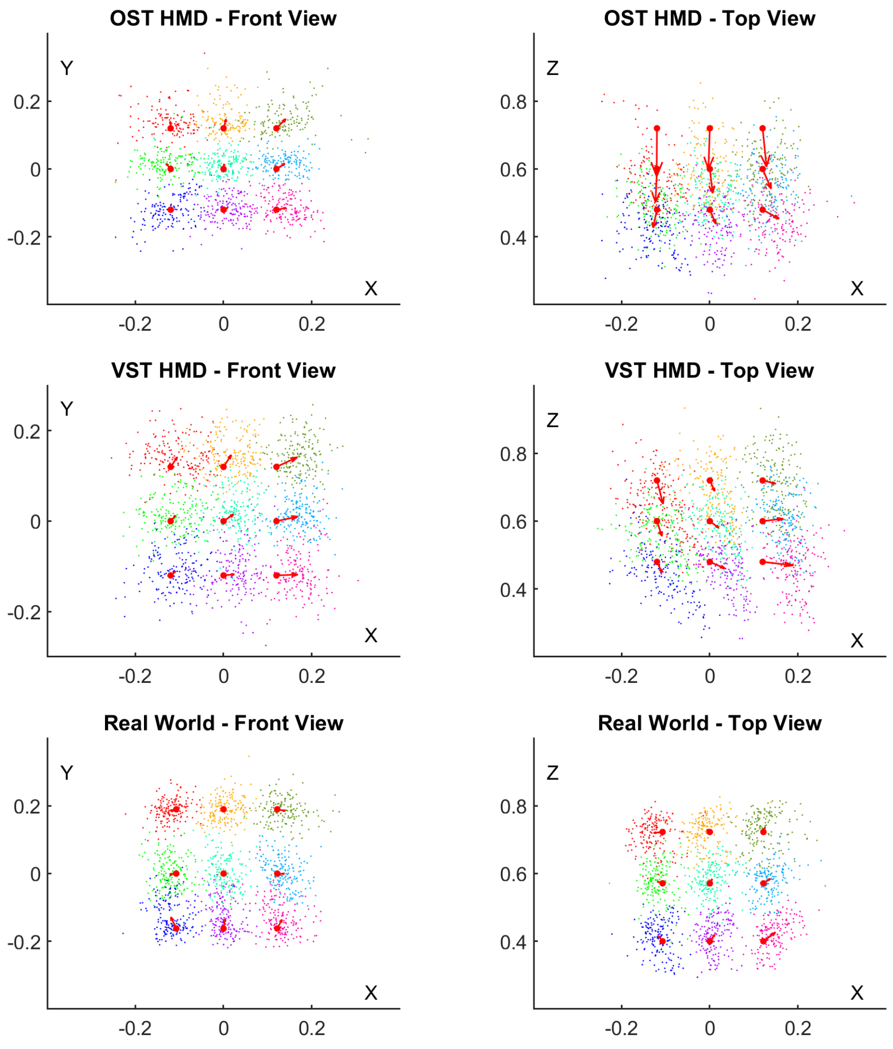


Figure 25: Results of the Blind Reaching Task (measures in meters). Red dots represent the stimulus position (target). Colored dots represents the user's fingertip positions during the blind reaching: a specific color is associated for each target position to see the related cluster of reaching positions. The vectors point the center of mass of the cluster, which encloses all the hits for that target. The origin is the user's point of view.

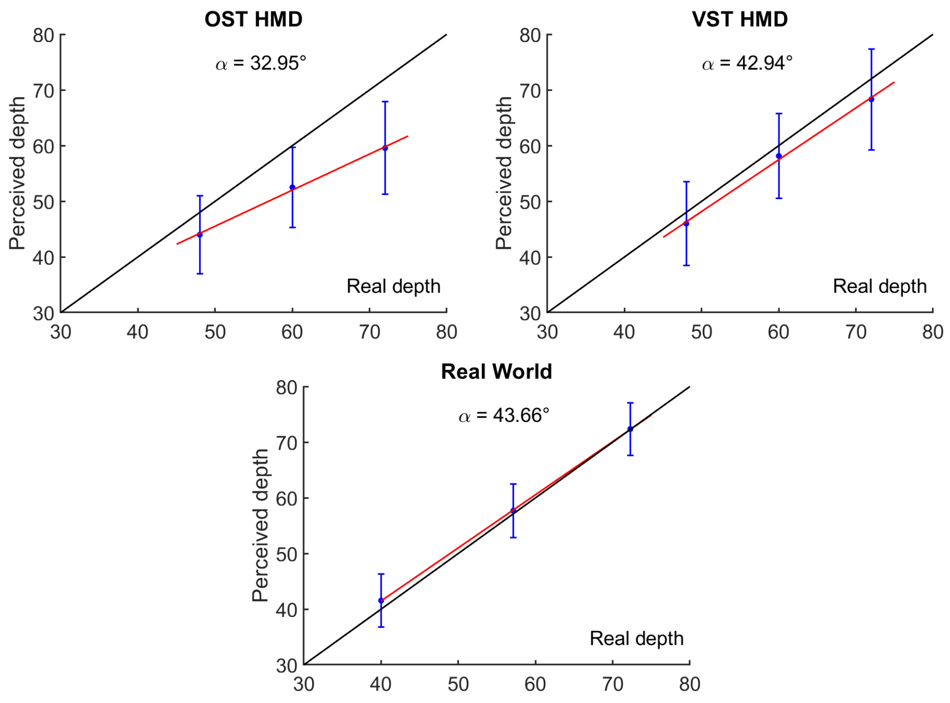


Figure 26: Linear fittings displaying the relationship between the real and perceived depth positions in the three conditions, where the bisector represents the ideal case (with equal real and perceived depth). The slope α in the OST group suggests that the undershooting, and thus compression of the perceived depth, grows as distance from the user increases.

4.3 Using Registered Devices with Feedback

This experiment aims at obtaining a comparison of the two AR devices in a functional and interactive task. The aim is to assess whether and how the distortions observed in the blind reaching experiment affect user interaction in more naturalistic settings, where multiple depth cues are available.

4.3.1 *Setup*

Subjects were asked to move a few common household items on a table and overlap them to their virtual projection (see Figure 27,28). The CAD models used for the rendering were not necessarily a replica of the real ones (e.g., our box of cereal was of a different brand). Still, the models' shapes and sizes were scaled accordingly to be dimensionally coherent with the real items used.

The virtual objects appeared in a 3×3 grid (20 cm distance between central points) on a table in front of the users. We choose to have a larger workspace area with respect to the previous experiment ($40x40$ cm instead of $24x24$ cm), increase the users' mobility around the scene, observe their behaviour and the possible undesired effects like sickness in a more naturalistic task. We fixed the rendering positions of the virtual objects on the 3x3 grid by placing a Vive Tracker on the 9 possible positions on the table and recording its position: the grid is therefore rendered in the same place for all the users. The subject had as much time as needed to align the real object with the corresponding virtual one, and no constraints were imposed on the movements. The virtual items' spawning position was randomized by shuffling the list of the 9 possible positions of the grid. Each user aligned 45 items, thus moving to the same point of the grid 5 times. Among all the users, virtual objects were displayed 60 times on every candidate position of the grid.

The experiment was conducted under normal lighting, thus providing all the cues which would be present for normal usage circumstances of the OST and VST HMDs. We gathered data from 12 volunteers (9 males, 3 females) who were evenly split into two groups of 6 people. Each group started the experiment with a different HMD to avoid bias (balanced within-subject experimental design). All the subjects had already performed the blind reaching experiment thus, they knew the calibration procedures. Of the original 15 participants from the blind reaching experiment, 3 could not attend the second experiment: the average age of the examined sub-group was 30 ± 7 years. The average height was 174 ± 9 cm.

In this experiment, the perceived position of the virtual objects was recorded through a Vive Tracker fixed on a glove (instead of the Kinect as in the previous experiment) to increase the freedom of interaction movements. This is also motivated by the fact that we do not need to detect the position of a finger, but just the position of the hand that grasps the real items. We built the glove with a 3D printed ankle band adapter attached with two Velcro straps on a custom adjustable glove, sewed from washable synthetic fabric. Data gloves, which also include precise finger tracking, are commercially available (e.g., Manus gloves) but not necessary for our experiment.

During this experiment, we asked the participants to fill the Simulator Sickness Questionnaire (SSQ) [52] and the Igroup Presence Questionnaire (IPQ) [76] to measure user experience differences between the two HMDs.

The SSQ consists of 16 questions, to be answered on 4 points Likert scale, indicating the level of perceived symptoms from None to Severe. The evaluation of simulator sickness for a human-computer interaction topic is still under discussion [83]. Still, the SSQ is the de facto standard to evaluate undesired effects when using VR and AR devices. To obtain an indication of the symptoms over the Nausea, Oculomotor, and Disorientation domains, the answers to the 16 questions are weighted as in [101].

The IPQ is a questionnaire for measuring the sense of presence experienced in a virtual environment. The IPQ has three subscales and one additional general item not belonging to any specific subscale. The three subscales are: *Spatial Presence*, i.e., the sense of being physically present in the virtual environment; *Involvement*, measuring the attention devoted to the environment; and *Experienced Realism*, measuring the subjective experience of realism in the virtual environment [82].

4.3.2 *Procedure*

At the beginning of the experiment, the subject is asked to fill the SSQ (PRE) questionnaire. The experimenter briefly explained the task. Then, after the same calibration procedure described in the previous experiment carried out, the task started. The experimenter spawned a virtual item on the table (see Figs. 2728). After the user aligned the real object with its virtual counterpart, the user was asked to keep the palm flat, above each object, centered on the object's midpoint along the XZ plane. The experimenter recorded the perceived location with a button press, and a new virtual item was generated. It is worth noting that to proceed with the alignment, the user can move the real objects present on the table, thus creating a new and not controlled configuration of the real objects. We ensured that the user's concept of the center was the same as the actual physical center by showing which position must be considered as the central one for each object before the start of the experiment. Moreover, all the objects used were either circular or mostly symmetrical, thus no misinterpretation of the task was likely possible. After the subject finished aligning 45 items, the task ended. The subject removed the HMD, and filled the IPQ, SSQ (POST) and SSQ (PRE, for the second trial) questionnaires. The same procedure was then repeated with the other HMD. After finishing the task again, the SSQ (POST) and IPQ questionnaires were filled. Since this experimental setup was much quicker than the blind reaching, we performed the test with the two HMDs consecutively, without letting the user's rest 1 day as in the first experiment. However, we can safely exclude any bias introduced by fatigue both because the first HMD used was equally different for the whole experimental group and because the starting user conditions were recorded before each test through the SSQ questionnaire (PRE). Two videos showing the First-person task experiment with the VST [1] and the OST [2] devices are available.

4.3.3 *Data Analysis*

In this experiment, the task is to move the real objects onto the virtual corresponding ones. To quantify the errors in performing the task, the position of the user's palm, after the alignment of the real object with the virtual one, is compared with the position of the displayed virtual object in the kitchen scenario (Figure 29). Each subject performed 45 alignments for a total of 540 interactions. The mean absolute error over the x-axis is $17 \pm 11mm$ for the OST HMD and $18 \pm 17mm$ for the VST HMD. For the z-axis (depth), the mean absolute error is $28 \pm 20mm$ for the OST HMD and $35 \pm 25mm$ for the VST HMD. The Wilcoxon Signed-Rank test shows no significant difference between the two HMDs over the x-axis. The depth errors (z-axis), on the other hand, are found to be different ($p < 5e^{-5}$), with the VST HMD slightly underperforming when compared to the OST HMD.

The SSQ scores (Figure 30) also display a better performance of the OST HMD, with the VST HMD increasing all the motion sickness symptoms significantly more. A significant increase in nausea, oculomotor and total scores are observed for the VST HMD, and no substantial increase in any score for the OST HMD. The cross validation between the starting conditions (OST-VST pre) of the subjects, who used different HMDs, show no significant difference between them. The cross validation between the final conditions (OST-VST post) shows a significant difference in the way the nausea and total scores increase during the use of the two different HMDs.

Finally, a Wilcoxon Signed-Rank test shows that the results of the IPQ questionnaire (Figure 31) display no significant differences in the perceived sense of presence between the users of the two HMDs.

4.3.4 *Outcome*

In this experiment, the users are able to use many more depth cues with respect to the blind case. The perspective of the scene from above (i.e. on the XY plane with respect the line of sight), together with the ability to move around, helps the user perform the task. The feedback exploited during the alignment task (i.e. the visual alignment cue) greatly enhances the user's accuracy and precision: the errors observed during the experiment are coherent with the errors measured in the control group of the blind reaching experiment. This behaviour is also coherent with the results of our previous study [10], where users performed significantly better when the visual feedback is possible (i.e. the visual alignment between the real and the virtual object) was provided during the movement. It must be noted how the movement, in this case, was

1 `https://youtu.be/kLTj-x8cSAw`
2 `https://youtu.be/2xpd1Q_jRlg`

constrained for simplicity on a plane (the table), but without loss of generalization, we expect to observe similar behaviours even while the same task is carried out at different heights. Similar experimental setups are not uncommon in depth perception experiments (e.g. in [53] a similar setup is proposed for VR depth perception). Our final consideration is that albeit current AR HMDs do indeed introduce enough aberrations that distort the user's spatial perception, it is possible to achieve an effective interaction in AR with enough visual feedbacks.

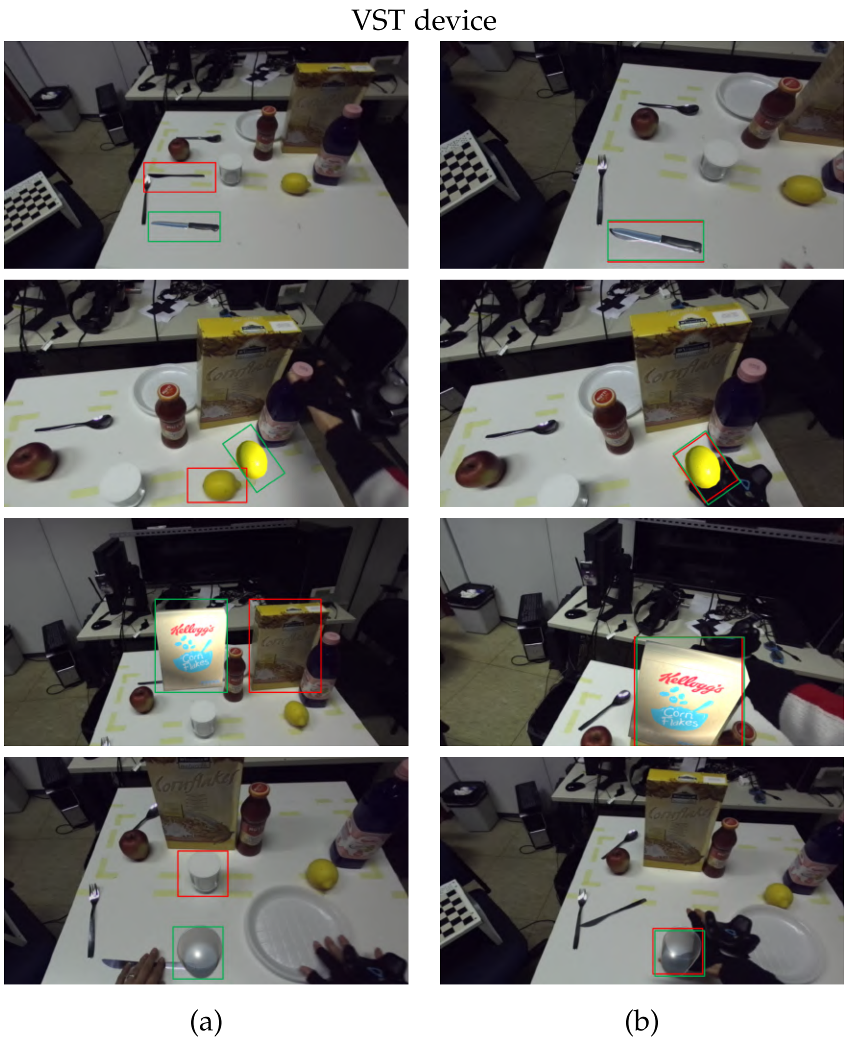


Figure 27: Frames of the interaction task experiment, for the VST device. The left column (a) shows the scene before the task: the real and virtual corresponding objects are highlighted with red and green boxes, respectively. The task consisted in moving the real objects onto the virtual corresponding ones. The right column (b) shows the scene after the task has been completed.

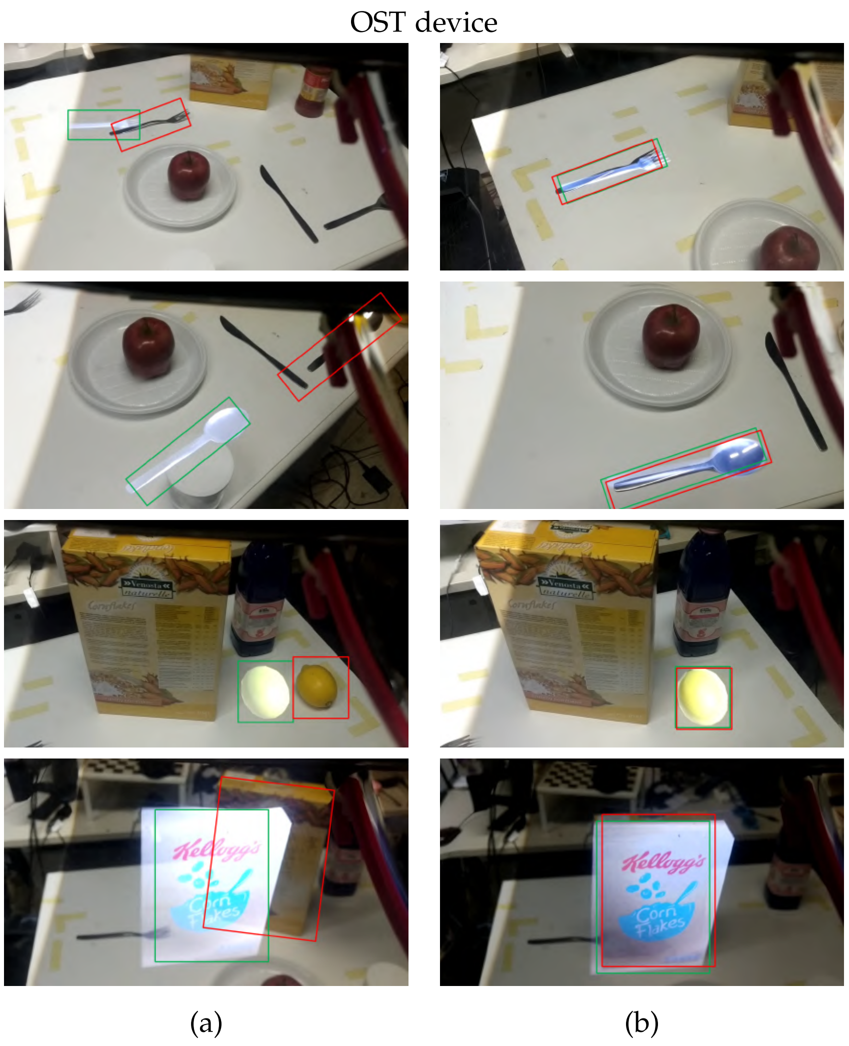


Figure 28: The same representation displayed in Figure 27, but for the OST HMD: before, column (a), and after the alignment, column (b).

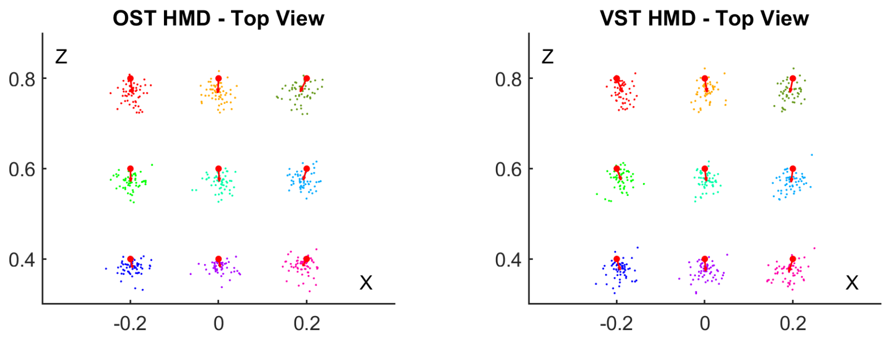


Figure 29: The hand positions (colored dots, as in Figure 25) recorded during the interaction task experiment with the OST and VST HMDs (measures in meters). Red dots represent the displayed object position. A small undershooting can be observed, which is likely due to the palm being slightly offset with respect to the center of an object (red dots) during a grasp. The vectors point to the center of each cluster. The origin is the user's point of view.

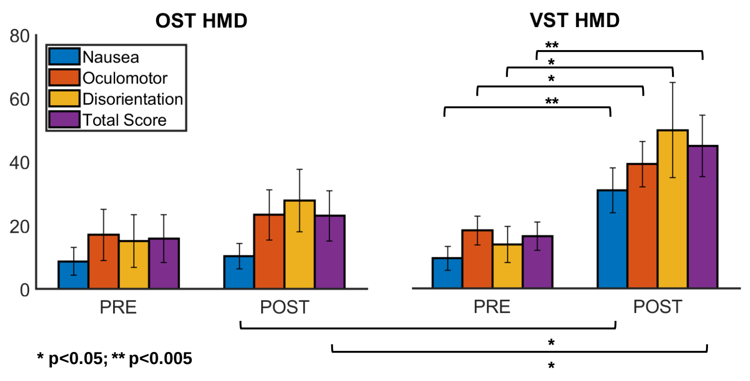


Figure 30: Interaction task experiment: the results of the SSQ questionnaire for the OST and the VST devices (left and right, respectively). Each plot shows the mean values, averaged on all the participants, and the associated standard deviation, for 3 subscales and the total score. The SSQ questionnaire has been filled before (PRE) and after the experiment (POST). The Wilcoxon Signed-Rank Test p-values are displayed where a significant difference is found (i.e. when the paired, two-sided test rejects the null hypothesis of zero median in the difference between paired samples at the 5% significance level).

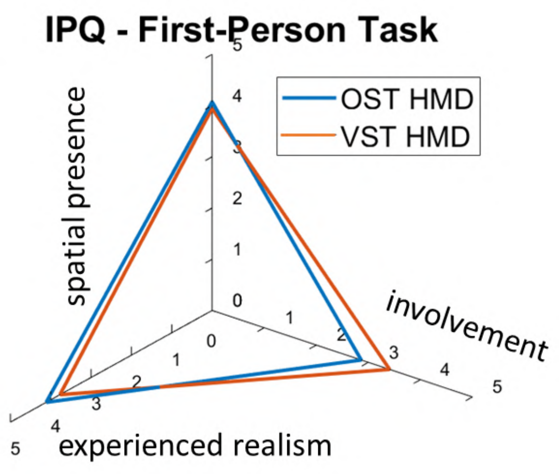


Figure 31: First-person interaction task experiment: the results of the IPQ questionnaire. Blue is for OST HMD, and red for VST HMD. The values, averaged across the participants, are represented for the 3 subscales.

5 Conclusions

The contribution of this thesis is twofold: first, we developed a registration framework which can be used to develop AR applications where real and virtual elements are co-localized and registered in a common reference frame. The generalized nature of the framework allows the use of different types of headsets, both VST and OST, and tracking systems, and provides a way to track and register the user's interaction in the same common reference frame where the virtual elements are displayed.

Moreover, we have used the proposed framework to quantitatively assess and compare the interaction and 3D position perception in the peripersonal space when using OST and VST HMDs, in rigorous experimental settings. Indeed, to perform a quantitative comparison, all the elements of the scene must be expressed with respect to the same, known, reference frame. This also applies to the users' fingers and hands, if the interaction is quantified as well.

The proposed framework is made of several modules, which are all integrated inside the Unity graphical engine. All the measures have been registered in the VST HMD reference frame, as it provides a robust way to precisely locate designated points inside a specified area, with IR tracking. The system can be easily adapted to work with similar systems, i.e. using other VST HMDs IR tracking systems (e.g. Oculus Rift one instead of HTC Vive system) or any other forms of tracking systems able to perform precise punctual tracking (e.g. MOCAP). To compare the performances of different types of HMDs, different devices have thus been registered in the VST HMD reference frame.

One module deals with the registration of a generic OST HMD: the visual tracking of the HMD has been disabled, to use the HMD merely as a visualization device, isolating any bias possibly introduced by the robustness of the manufacturer SLAM system. The HMD can thus be tracked with an external tracker, which is being tracked in the VST HMD reference frame, and then calibrated with the SPAAM technique implemented in the module to obtain a generic profile which adjusts the projection to avoid the parallax introduced by the mismatch between the location of the tracker with respect to the user's eyes optical center position. The module is designed to obtain a baseline calibration profile first, as in the literature the calibration procedure has been proven to be very susceptible to user's error. The baseline profile can be obtained by performing several calibrations in a strict setting, using a specifically constructed mannequin with a stereo camera resembling the human's eyes. In this way, the alignment error caused by the user's sway (i.e. caused by breathing or by operating the confirmation method) is kept under control. Rather than moving the mannequin head to perform the alignment, the target can be moved instead in fixed stable positions during the alignment phase. To further reduce the calibration error, the calibration module implements a RANSAC procedure which removes outliers in over-determined systems, i.e., whenever more alignments than the strict minimum needed six are obtained.

The final alignment performed by each user before using the HMD, to deal with the residual calibration error, is dealt with another module. The users align a virtual checkerboard with a real one, which is also being tracked in the VST HMD reference system. A visual gizmo is displayed, and the alignment is performed through verbal communication with the experimenter, which adjusts the projection as communicated by the user. The module adjusts the parameters of the virtual cameras which are used by the graphic engine to generate the final projection rendering in tiny steps until a correct alignment is achieved.

To isolate the baseline error, i.e. the measured error which is not caused by a perceptual mismatch, a control group is required. One module of the frameworks is a setup that can be used to measure the baseline error, i.e. the error measured when the task is performed in non-AR settings, which is thus caused by other factors such as the sum of all the registration errors of the various tracking devices. The proposed setup is able to provide visual stimuli in different 3D positions which are identical to the ones perceived with AR.

Finally, the last modules of the framework track the user's finger position and register it in the same common reference frame as all the other devices. The user's finger position is tracked with a color segmentation algorithm which can be configured to use different HSV values to track different colors as needed, with a Microsoft Kinect V2. The Kinect is registered with an external Vive Tracker through an external program which is synchronized through an UDP connection, thus the system can be adapted to use not only other RGB-D sensors, but other types of tracking devices (e.g. Optitrack) as well.

To recap, the framework is able to (i) track the user's position in a common reference frame regardless of the used HMD; (ii) track the user's fingertip, which is also registered in the same common referance frame; (iii) track objects in the real scene to display virtual content in specified, repeteable positions. Finally, it provides a way to display the same visual stimuli perceived in AR without the need of an HMD, for validation purposes. The framework thus enables the development of complex AR applications, where the user is able to interact with virtual imagery which can be precisely displayed in chosen positions with respect to the real world. We thus exploited the framework to perform further studies on the perception/interaction loop, by carrying out several experiments.

The performed experiments investigate the perceptual differences between different AR visualization devices in peripersonal space. In the state of the art, several studies have been performed on spatial perception in AR, but many focus on mid to far distances perception. We chose to focus on peripersonal space perception instead, and thus on the ability to achieve an effective interaction with manipulative tasks (e.g. grasping, pointing). Indeed, for a proper interaction in close distances, the virtual augmentation must be particularly stable and devoid of optical aberrations. In this work, the aim is specifically to measure the differences between two different technological approaches to construct an AR HMD, i.e., using a VST model or an OST model. Due to their design differences, which have different implications on the user-HMD optical models, different behaviours have indeed been observed.

We first tested the differences between the devices in a stock, unregistered state. We took in consideration two different HMDs. We performed two tests, and compared the results between the subjects who used an Optical See-Through (OST) HMD (Metavision Meta 2) and those who used a Video See-Through (VST) HMD (a smartphone in conjunction with a headset like the Google Cardboard). The data has been collected from a total of 45 volunteer participants. In the experiment, the subjects had to perform a precision reaching task by overlapping the hand on the perceived target position. Then, we observed how the presence or absence of an internal feedback influenced the homing performance. Our results revealed a better depth estimation, thus a more precise interaction, when using the OST device, which also revealed a lower impact on eye strain and fatigue. As expected, the lack of a proper synchronization between the used devices, caused a misalignment between the measures obtained internally (through the HMD's sensors) and externally (through the Kinect), potentially introducing a bias in the results. We thus used the developed framework to perform a systematic comparison, by carrying out two experiments.

The first one is a 3D blind reaching experiment, where simple objects (spheres) appeared randomly in a 3D grid in front of the user. The results reveal a consistent underestimation ($\sim$ 10 cm) of depth (i.e. along the z-axis that is orthogonal to the viewer image plane) when using the OST device, and also differences between OST and VST devices along the x and the y-axis (left/right and up-/down directions with respect to the user). The baseline data, recorded by performing the same task without using HMDs, i.e. the real world condition, revealed that the task is usually performed with errors of less than 4 cm in depth. This result is interesting per se, since it provides a baseline of the user's performance in real condition, i.e. without wearing an HMD. The AR devices affect user's perception and thus the interaction capabilities, in line with similar results in the literature.

The second experiment considers a realistic interaction task in AR: people should move objects in a scene containing both real and virtual elements. In this case, the results reveal negligible (for the purposes of the given task) differences between VST and OST devices in terms of accuracy. Interestingly, such a result is in line with the previous result of the blind reaching task in real condition. Indeed, the egocentric geometry of the scene and the specific task make the 3D position perception constrained on the table, which is on the XY plane with respect to the observer's line of sight, and in the XY plane no appreciable differences between VST and OST are observed. A further observation is that the VST device causes more sickness than the OST one, this fact is due to the latency in the video stream and is coherent with other findings in the literature.

The conclusion of the performed experiments is that albeit spatial aberration and depth compression do seem to be relevant across commercial AR HMDs, with a proper feedback the user is able to achieve a functional interaction, which is within the bounds of the errors observed in the real world. Being able to have a visual feedback of one's hands (or other interaction mediums, e.g., controllers) in the HMD system reference frame allows the user to effectively adjust the residual registration error caused by the misalignment of the real and tracked environment. The need of this re-alignment process does however have a non-trivial cognitive load which leads to the development of different degrees of simulation sickness symptoms.

Our results do not solve the problem of choosing between VST and OST devices, nor was this the aim of the study. Rather, this study serves as a baseline for further studies by highlighting the specificity of egocentric perception in interactive AR and providing the proposed registration framework to the research community, as a set of independently working modules which can easily be adapted to work with any type of HMD or external tracking device.

6 Perspective

The results obtained lay the foundations for further researches both on the way humans perceive information coming from the surrounding environment and on the interaction with the environment, which are examined in this last part.

Augmented Reality has stimulated the fantasy of entire generations of people far before its birth in the 90's: first depictions of holograms started to appear in science fiction as far as in the fifties, from Isaac Asimov novels. Today, we are starting to see the spread of AR in many fields, but we are still far from achieving what is commonly depicted in futuristic settings in movie theaters: full blown out holograms animated thorough the city, and people seamlessly interacting with super advanced machines with holographic interfaces fully integrated with natural language processing. At the moment, AR devices still often struggles to put together all the technological and cognitive discoveries required to obtain stable enjoyable experiences for everyone. The environment must be properly tracked, segmented and labeled in real-time, and the user position in the environment must be known. Whenever multiple users are present in the scene, all of them must be registered with each other, for shared AR to be possible. The user interaction must be tracked in some way, which is starting to become possible thanks to real time hand pose estimation techniques, which are by themselves a very complex branch of the computer vision research field (e.g. due to their fast movement, tendency to self-occlusion, complex kinematic model, differences in skin colors, and so on). The devices themselves face several technological constraints, due to the difficulty of the optical system, the processing power required, the energy consumption, the limited field of view and frame rate. The future incarnation of AR will probably be through a mix of handheld and HMD devices, which, in the current form, have shown to be perceptually inaccurate with respect to the human sight system. The proposed work is a step forward in that direction, offering interesting perspectives in the development of both new devices and new applications.

The work conducted on the registration framework, for example, gives us a starting ground to perform further comparisons between different devices in strict experimental settings. Without properly synchronized and registered devices it would be impossible to perform a coherent analysis. Due to the generic nature of the framework, composed by several interlinked modules, it will be possible to add new devices (e.g. new HMDs, or handheld) for even broader comparisons. The user tracking module can be enriched to detect more complex interactions, i.e., by registering external hand trackers (like the Leap Motion) or by adding a routine to directly use the HMDs video stream (e.g. with MediaPipe). In our setup, the environment is tracked through IR trackers to achieve a better stability, but it is possible to increase the spatial awareness of the system by registering and modelling the surrounding environment (i.e. with similar approaches to the work discussed in [98]).

The experiments performed on the perception/interaction loop confirm the presence of optical aberrations in the devices used and give insight on the design of further similar experiments. The natural extension to the experiments performed is to assess if other HMDs have comparable performances with respect to the ones used in our setup, to eventually isolate more precisely the factors which have a greater impact on the achievement of a natural interaction (i.e. using the stock Vive Pro frontal cameras rather than the frontal mounted ZED mini stereo camera).

If mechanically achieving a parallax-free optical model is proven to be too difficult, and the spatial aberrations inevitable, a possible future approach could be to act on the rendering pipeline to lessen the effect of spatial compression that are currently observed in AR environment, e.g. non-linearly modifying the objects rendering position depending on the user position (in a similar way to haptic retargeting techniques, e.g. as seen in [5]).

Finally, some other interesting developments could be the investigation of other cognitive aspects behind AR UX design besides spatial perception, i.e., the comparison of AR user interface performances with respect to traditional interfaces, in terms of commonly evaluated indexes (e.g. learnability, memorability, or safety). Ergonomics and bio-mechanical factors might also be considered to develop a new set of UX guidelines, i.e., consider both the user's arms interaction volume and muscle load to split the peripersonal space in sub-volumes with different strain indexes, to be able to place items subject of frequent interactions in places associated with a lower impact on the user's strain.

Framework Documentation

The following sections provide the documentation with the technical details of the devised registration framework, which is made publicly available, licensed under *The Unlicense* (a license with no conditions whatsoever which dedicates works to the public domain): unlicensed works, modifications, and larger works may be distributed under different terms and without source code.

Appendix A shows the hierarchical index, which displays the inheritance list of the project's classes; Appendix B includes the class list with a brief description, and Appendix C includes the detailed description of each class, with their relative attributes and methods.

It must be noted that since Unity uses the entity/component model instead of the classic object-oriented programming (OOP) paradigm, all the classes actually inherit from the same class (MonoBehaviour, the base class from which every Unity script derives). For this reason, instead of an UML class diagram, the structure of the Unity scenes (and the relative entity/component relationships) are displayed in the provided repository, together with the suggested settings for each parameter. Nevertheless, the hierarchical index (Appendix A) has still been included as it can act as a compact index to quickly find the reference to the needed classes.

The framework repository is available on GitHub [6].

A

Hierarchical Index

A.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

B Class Index

B.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

C
Class Documentation

C.1 Checkerboard_tracker_pose Class Reference

Synchronizes the virtual checkerboard to the real one, updating the pose depending on the pose of the Vive Tracker attached to the real checkerboard. The rigid transformation between the Vive Tracker and the checkerboard has to be set inside Unity.

Inherits MonoBehaviour.

C.2 destroyTimer Class Reference

Destroys the associated gameObject after a set amount of time.

Inherits MonoBehaviour.

Public Attributes

- float lifetime

C.2.1 Member Data Documentation

lifetime `float destroyTimer.lifetime`

Time in seconds before the object destruction.

C.3 disableSteamVR Class Reference

Disables steamVR from automatically launching when the scene starts. Can be attached to any active gameObject of the scene.

Inherits MonoBehaviour.

C.4 enableCalibration Class Reference

Can be used to when the SPAAM calibration procedure is not needed, to disable all the components involved in the SPAAM calibration. Attach to a random gameObject and toggle disableCalibration to skip the SPAAM calibration procedure.

Inherits MonoBehaviour.

Public Attributes

- bool disableCalibration
- GameObject leftCamera
- GameObject rightCamera
- GameObject calibrationParameters

C.4.1 Member Data Documentation

calibrationParameters `GameObject enableCalibration.calibrationParameters`
gameObject used to display the computed parameters inside the Unity editor.

disableCalibration `bool enableCalibration.disableCalibration`
If set to true, the crosshair shaders applied to the virtual cameras will be disabled, and the SPAAM procedure will not start.

leftCamera `GameObject enableCalibration.leftCamera`
gameObject of the left camera contained in the metaCameraRig prefab.

rightCamera `GameObject enableCalibration.rightCamera`
gameObject of the right camera contained in the metaCameraRig prefab.

C.5 Hmd_tracker_pose Class Reference

Updates the head position, depending on the pose of the Vive Tracker attached to the HMD. The rigid transformation between the Vive Tracker and the HMD has to be set inside Unity (after obtaining it through calibration).

Inherits MonoBehaviour.

C.6 IdentifyTrackerID Class Reference

Automatically associates the correct ViveTracker to a specific gameObject at runtime, without the need to select the active device.

Inherits MonoBehaviour.

Public Attributes

- string trackerId

C.6.1 Member Data Documentation

trackerId `string IdentifyTrackerID.trackerId`
Id of the tracker, obtainable from steamVR->handle Vive Trackers

C.7 KeepBetweenScenes Class Reference

Used to avoid the need to reinitialize the HMD/calibrate again on scene change.
Inherits MonoBehaviour.

C.8 NeutralizeRotation Class Reference

Neutralizes any translation/rotation/scaling of a child GameObject by applying the inverse transformation.
Inherits MonoBehaviour.

Public Member Functions

- void **LateUpdate** ()

C.8.1 Detailed Description

In order to use this class to neutralize tracking of the Meta headset, create a new empty GameObject and place this script on it. Then make the Meta↩ CameraRig GameObject the only child of this new GameObject. You can then parent this GameObject by a further empty GameObject on which you can apply any custom transformation (such as third-party tracking).

C.9 packData Class Reference

Fetches the data which can then be sent through a socket to another project.
Inherits MonoBehaviour.

Public Types

- enum class **Tracker** { **HMD** , **CalibrationTarget** , **Checkerboard** }

Public Attributes

- Tracker selectedTracker

C.9.1 Detailed Description

It is assumed to have 3 tracked objects: the Vive Tracker on the HMD, the Vive Tracker attached to the checkerboard calibration pattern and the Vive Tracker used as calibration target during the SPAAM procedure. Change the name of the gameObject accordingly to change the items which are sent through the socket. First, create a prefab, e.g. "Synced_Data" which as a gameObject for each tracked object in the scene. Then, attach this script to each tracked gameObject. Select the corresponding associated gameObject in the dropdown menu created by the public enumerator. The created prefab must then be present in both Unity projects to be able to sync them through the socket connection. Add the created prefab as the "Player Prefab" inside the Spawn Info of the Network Manager class.

C.9.2 Member Data Documentation

selectedTracker `Tracker packData.selectedTracker`
Used to display a dropdown menu inside the Unity editor, to correctly associate each tracker to their correct gameObject.

C.10 reachingTaskController Class Reference

Inherits MonoBehaviour.

C.11 relPosTest Class Reference

Can be used to display the relative position between two gameObjects.
Inherits MonoBehaviour.

Public Attributes

- GameObject b
- Vector3 relPos

C.11.1 Detailed Description

The script must be attached to the first gameObject, while the public variable "b" needs to be set to the second gameObject. The relative position between the two objects will be displayed in the editor through the relPos public variable.

C.11.2 Member Data Documentation

b `GameObject relPosTest.b`

Add the second gameObject, of unknown relative position with respect with the master gameObject.

relPos `Vector3 relPosTest.relPos`

Output relative position from the master gameObject to the chosen gam↩ Object.

C.12 savedParameters Class Reference

Used to display the parameters computed through SPAAM calibration.

Inherits MonoBehaviour.

Public Attributes

- Vector3 leftEyePosition
- Quaternion leftEyeRotation
- Vector3 rightEyePosition
- Quaternion rightEyeRotation
- double SensorSizeX_left
- double SensorSizeY_left
- double LensShiftX_left
- double LensShiftY_left
- double SensorSizeX_right
- double SensorSizeY_right
- double LensShiftX_right
- double LensShiftY_right
- bool leftEyeCalibrated = false

C.12.1 Detailed Description

Attach this script to an empty gameObject. If the enableCalibration class is used in the scene, this gameObject must be passed to its "calibrationParameters" public variable.

C.12.2 Member Data Documentation

leftEyeCalibrated `bool savedParameters.leftEyeCalibrated = false`

Used to internally track which eye has been calibrated.

leftEyePosition `Vector3 savedParameters.leftEyePosition`

Position of the left eye with respect to the device origin. e.g., position which needs to be set as the left camera inside the MetaCameraRig prefab.

leftEyeRotation `Quaternion savedParameters.leftEyeRotation`
Rotation of the left eye with respect to the device origin. e.g., rotation which needs to be set as the left camera inside the MetaCameraRig prefab.

LensShiftX_left `double savedParameters.LensShiftX_left`
Computed principal point offset, on the x axis, for the left eye.

LensShiftX_right `double savedParameters.LensShiftX_right`
Computed principal point offset, on the x axis, for the right eye.

LensShiftY_left `double savedParameters.LensShiftY_left`
Computed principal point offset, on the y axis, for the left eye.

LensShiftY_right `double savedParameters.LensShiftY_right`
Computed principal point offset, on the y axis, for the right eye.

rightEyePosition `Vector3 savedParameters.rightEyePosition`
Position of the right eye with respect to the device origin. e.g., position which needs to be set as the right camera inside the MetaCameraRig prefab.

rightEyeRotation `Quaternion savedParameters.rightEyeRotation`
Rotation of the right eye with respect to the device origin. e.g., rotation which needs to be set as the left camera inside the MetaCameraRig prefab.

SensorSizeX_left `double savedParameters.SensorSizeX_left`
Computed sensor size width for the left eye.

SensorSizeX_right `double savedParameters.SensorSizeX_right`
Computed sensor size width for the right eye.

SensorSizeY_left `double savedParameters.SensorSizeY_left`
Computed sensor size height for the left eye.

SensorSizeY_right `double savedParameters.SensorSizeY_right`
Computed sensor size height for the right eye.

C.13 savePositionsToFile Class Reference

Simple script used to save the 3D positions of the points tracked by the Kinect Registration Module, and the 3D positions of a Vive Tracker, to file.
Inherits MonoBehaviour.

Public Attributes

- string logDataFilename ="realPositions"

C.13.1 Detailed Description

Used to verify the Kinect registration accuracy and precision by comparing the 3D world coordinates obtained with a Vive tracker with respect to the same positions tracked by the Kinect V2. The assigned key-bindings for the Rotation editing mode are the following ones:

- T: Saves the Vive tracker position, in world coordinates, to file.
- K: Saves the Kinect tracked point, in world coordinates, to file.
- O: Finishes the session and closes the log file. It is then possible to exit the editor.

C.13.2 Member Data Documentation

logDataFilename `string savePositionsToFile.logDataFilename ="realPositions"`
Filename where the recorded data txt will be saved.

C.14 socketReceiver Class Reference

Inherits MonoBehaviour.

Public Member Functions

- void **Start** ()
- string **getLatestUDPPacket** ()

Public Attributes

- int port
- float xKinectTracked = 0
- float yKinectTracked = 0
- float zKinectTracked = 0

C.14.1 Member Data Documentation

port `int socketReceiver.port`
Port used for the transmission. Must be the same of the one used in the Kinect Registration Module (default is 8888).

xKinectTracked `float socketReceiver.xKinectTracked = 0`
x coordinate of the 3D point tracked by the Kinect, expressed with respect to the Kinect reference frame. Visualization only.

yKinectTracked `float socketReceiver.yKinectTracked = 0`
y coordinate of the 3D point tracked by the Kinect, expressed with respect to the Kinect reference frame. Visualization only.

zKinectTracked `float socketReceiver.zKinectTracked = 0`
z coordinate of the 3D point tracked by the Kinect, expressed with respect to the Kinect reference frame. Visualization only.

C.15 SPAAM Class Reference

Single Point Active Alignment Method (SPAAM) implementation for the OST HMD.
Inherits MonoBehaviour.

Public Attributes

- bool calibrateRightEye
- float disparity
- bool enableLogging
- bool enableDebugLog
- bool enableRANSAC
- double inlierDistanceThreshold
- int ransacPointsPerBatch
- float maxError
- bool **useIntrinsics**
- bool useMyInputParameters
- List< float > myTrackerX = new List<float>()
- List< float > myTrackerY = new List<float>()
- List< float > myTrackerZ = new List<float>()
- List< float > myU = new List<float>()
- List< float > myV = new List<float>()
- float adjustmentShift = 0
- float adjustmentAngle = 0
- int resolutionWidth
- int resolutionHeight
- float pixelSizeX = 124/2560/1000
- float pixelSizeY = 85.5f/1440/1000
- List< float > xSpawnPositionsUserDefined = new List<float>()
- List< float > ySpawnPositionsUserDefined = new List<float>()
- int numberOfMatches
- int spriteSize
- GameObject viveTracker
- GameObject viveCamera
- Matrix4x4 originalProjection

- bool flipX
- bool flipY
- bool flipZ
- bool Normalization
- bool metricConversion
- bool removeCrosshairShift
- bool usePixelSizes

C.15.1 Member Data Documentation

adjustmentAngle `float SPAAM.adjustmentAngle = 0`
Legacy parameter used for the implementation.

adjustmentShift `float SPAAM.adjustmentShift = 0`
Legacy parameter used for the implementation.

calibrateRightEye `bool SPAAM.calibrateRightEye`
Set to true to start the calibration with the right eye. By default the procedure starts by calibrating the left eye.

disparity `float SPAAM.disparity`
Disparity (in mm) between the two LCDs of the OST HMD.

enableDebugLog `bool SPAAM.enableDebugLog`
Set to true to generate a more verbose log.

enableLogging `bool SPAAM.enableLogging`
Set to true to generate a log with the computations.

enableRANSAC `bool SPAAM.enableRANSAC`
Set to true to use the RANSAC to remove outliers from the set of performed alignments.

flipX `bool SPAAM.flipX`
Set to true if the image x-axis and camera x-axis point in opposite directions.

flipY `bool SPAAM.flipY`
Set to true if the image y-axis and camera y-axis point in opposite directions.

flipZ `bool SPAAM.flipZ`
Set to true if the camera looks down the negative-z axis.

inlierDistanceThreshold `double SPAAM.inlierDistanceThreshold`
Threshold used to define the maximum reprojection error of an alignments which are considered inliers.

maxError `float SPAAM.maxError`
Defines the stopping criteria of the RANSAC procedure. When the likelihood of finding a better model becomes lower than this parameter, the best model found so far is saved.

metricConversion `bool SPAAM.metricConversion`
Set to true to convert the coordinates of the shown crosshair in metric form, using Pixel Size (x,y) as parameters.

myTrackerX `List<float> SPAAM.myTrackerX = new List<float>()`
List of the x positions (in meters) of the SPAAM target during the alignment task (Xw)

myTrackerY `List<float> SPAAM.myTrackerY = new List<float>()`
List of the y positions (in meters) of the SPAAM target during the alignment task (Yw)

myTrackerZ `List<float> SPAAM.myTrackerZ = new List<float>()`
List of the z positions (in meters) of the SPAAM target during the alignment task (Zw)

myU `List<float> SPAAM.myU = new List<float>()`
List of the x positions (in pixels) of the crosshair displayed during the alignment task

myV `List<float> SPAAM.myV = new List<float>()`
List of the y positions (in pixels) of the crosshair displayed during the alignment task

Normalization `bool SPAAM.Normalization`
Set to true to normalize the coordinates of the shown crosshair between -1 and 1.

numberOfMatches `int SPAAM.numberOfMatches`
Number of matches required for the SPAAM procedure for each eye. The minimum should be at least 6, but since RANSAC is being implemented, more is advised.

originalProjection `Matrix4x4 SPAAM.originalProjection`
Variable used to display the original 4x4 projection matrix (before calibration).

pixelSizeX `float SPAAM.pixelSizeX = 124/2560/1000`
Size of the projected pixel width of the OST HMD.

pixelSizeY `float SPAAM.pixelSizeY = 85.5f/1440/1000`
Size of the projected pixel height of the OST HMD.

ransacPointsPerBatch `int SPAAM.ransacPointsPerBatch`
Number of alignments processed during each RANSAC iteration. The minimum number of alignments needed to obtain a well-defined system is 6.

removeCrosshairShift `bool SPAAM.removeCrosshairShift`
Set to true to perform the alignment with respect to the bottom left corner rather than the crosshair centre.

resolutionHeight `int SPAAM.resolutionHeight`
Height resolution (in pixel) of the OST HMD LCD display used for the projection on the lenses.

resolutionWidth `int SPAAM.resolutionWidth`
Width resolution (in pixel) of the OST HMD LCD display used for the projection on the lenses.

spriteSize `int SPAAM.spriteSize`
Size of the sprite used as crosshair during the alignment task. The sprite is assumed to be square. e.g. 64 means a 64x64pixel wide sprite.

useMyInputParameters `bool SPAAM.useMyInputParameters`
Set to true to use custom input parameters for the SPAAM procedure (for offline debugging)

usePixelSizes `bool SPAAM.usePixelSizes`
Set to true to convert u,v image plane coordinates into metric units.

viveCamera `GameObject SPAAM.viveCamera`
The GameObject which contains the calibrating camera.

viveTracker `GameObject SPAAM.viveTracker`
GameObject which must be set as child of the HMD camera and copy the 3D tracked world point coordinates for each update step.

xSpawnPositionsUserDefined `List<float> SPAAM.xSpawnPositionsUserDefined = new List<float>()`
List of the x positions (in pixels) where the crosshair will be displayed during the alignment task

ySpawnPositionsUserDefined `List<float> SPAAM.ySpawnPositionsUserDefined = new List<float>()`

List of the y positions (in pixels) where the crosshair will be displayed during the alignment task

C.16 SpawnTargets Class Reference

Handles most of the logic behind the blind reaching OST HMD experimental setup proposed.

Inherits MonoBehaviour.

Public Attributes

- int trialsNumber = 0
- string logDataFilename = "data"
- int depth = 3
- int width = 3
- int height = 3
- float spawnDistance = 5
- float userDistance = 50
- GameObject Volume
- GameObject Head
- GameObject target
- float targetSize = 1

C.16.1 Member Data Documentation

depth `int SpawnTargets.depth = 3`

Number of possible depth spawn positions in the interaction area. Distance between spawn points defined by SPAWN DISTANCE parameter.

Head `GameObject SpawnTargets.Head`

GameObject of the ViveTracker which tracks the OST HMD.

height `int SpawnTargets.height = 3`

Number of possible height spawn positions in the interaction area. Distance between spawn points defined by SPAWN DISTANCE parameter.

logDataFilename `string SpawnTargets.logDataFilename = "data"`

Filename where the recorded data will be saved. An increasing index will be added at the end, e.g. writing "test_" here will generate "test_0.txt", "test_1.txt" and so on

trialsNumber `int SpawnTargets.trialsNumber = 0`
The total number of stimuli that will displayed to the user during the whole experiment. In our study we used 27, but it can be changed.

spawnDistance `float SpawnTargets.spawnDistance = 5`
Minimum distance between spawn positions (centimeters).

target `GameObject SpawnTargets.target`
Prefab of the displayed stimuli (in our case, a circle).

targetSize `float SpawnTargets.targetSize = 1`
Scale of the prefab used to display the stimuli. Will be applied as invariant scale factor.

userDistance `float SpawnTargets.userDistance = 50`
Distance of the volume from the user (centimeters). Measured from the centre of the volume.

Volume `GameObject SpawnTargets.Volume`
Root gameObject used as bounding volume for the stimuli grid: all the stimuli prefabs will be instantiated as children of this gameObject. Create an empty GameObject "Volume" and position it as needed in the scene.

width `int SpawnTargets.width = 3`
Number of possible width spawn positions in the interaction area. Distance between spawn points defined by SPAWN DISTANCE parameter.

C.17 spawnTargetsExp2 Class Reference

Handles most of the logic behind the active task OST HMD experimental setup proposed.
Inherits MonoBehaviour.

Static Public Member Functions

- static List< int > **GenerateRandom** (int count, int min, int max)

Public Attributes

- int trialsNumber = 0
- string logDataFilename = "data"
- List< Vector3 > spawnPosition = new List<Vector3>(9)
- List< int > spawnSequence = new List<int>()
- List< GameObject > kitchenObjects
- GameObject Head

C.17.1 Member Data Documentation

Head `GameObject spawnTargetsExp2.Head`
GameObject of the ViveTracker which tracks the OST HMD.

kitchenObjects `List<GameObject> spawnTargetsExp2.kitchenObjects`
List filled with all the different objects models available. Every time a new object is spawned, a random one will be picked from the added library of objects.

logDataFilename `string spawnTargetsExp2.logDataFilename = "data"`
Filename where the recorded data will be saved. An increasing index will be added at the end, e.g. writing "test_" here will generate "test_0.txt", "test_1.txt" and so on

trialsNumber `int spawnTargetsExp2.trialsNumber = 0`
Number of times the entire grid will be displayed in a randomized order: we used 5. Change at your own discretion.

spawnPosition `List<Vector3> spawnTargetsExp2.spawnPosition = new List<Vector3>(9)`
Define the x, y, z coordinates where the items will be spawned. Measure the x, y, z coordinates by placing a vive tracker in the desired positions and saving the Vive Tracker Coordinates.

spawnSequence `List<int> spawnTargetsExp2.spawnSequence = new List<int>()`
Can be used to define a custom sequence.If left undefined, the vector of positions will be shuffled and a target will spawn into a random position until every item of the grid has been displayed. The process is repeated for several times (defined by "trialsNumber" in the Experimental Settings).

C.18 spawnTargetsExp2Vive Class Reference

Handles most of the logic behind the active task VST HMD experimental setup proposed.
Inherits MonoBehaviour.

Static Public Member Functions

- static List< int > **GenerateRandom** (int count, int min, int max)

Public Attributes

- int trialsNumber = 0
- string logDataFilename ="data"
- List< Vector3 > spawnPosition = new List<Vector3>(9)
- List< int > spawnSequence = new List<int>()
- List< GameObject > kitchenObjects
- GameObject Head

C.18.1 Member Data Documentation

Head `GameObject spawnTargetsExp2Vive.Head`
GameObject of the HMD prefab of the VST HMD.

kitchenObjects `List<GameObject> spawnTargetsExp2Vive.kitchenObjects`
List filled with all the different objects models available. Every time a new object is spawned, a random one will be picked from the added library of objects.

logDataFilename `string spawnTargetsExp2Vive.logDataFilename ="data"`
Filename where the recorded data will be saved. An increasing index will be added at the end, e.g. writing "test_" here will generate "test_0.txt", "test_1.txt" and so on

trialsNumber `int spawnTargetsExp2Vive.trialsNumber = 0`
Number of times the entire grid will be displayed in a randomized order: we used 5. change at your own discretion.

spawnPosition `List<Vector3> spawnTargetsExp2Vive.spawnPosition = new List<Vector3>(9)`
Define the x, y, z coordinates where the items will be spawned. Measure the x, y, z coordinates by placing a Vive tracker in the desired positions and saving the Vive tracker Coordinates.

spawnSequence `List<int> spawnTargetsExp2Vive.spawnSequence = new List<int>()`
Can be used to define a custom sequence.If left undefined, the vector of positions will be shuffled and a target will spawn into a random position until every item of the grid has been displayed.The process is repeated for several times (defined by "trialsNumber" in the Experimental Settings).

C.19 SpawnTargetsVive Class Reference

Handles most of the logic behind the blind reaching VST HMD experimental setup proposed.

Inherits MonoBehaviour.

Public Attributes

- int trialsNumber = 0
- string logDataFilename = "data"
- int depth = 3
- int width = 3
- int height = 3
- float spawnDistance = 5
- float userDistance = 50
- GameObject Volume
- GameObject Head
- GameObject target
- float targetSize = 1

C.19.1 Member Data Documentation

depth `int SpawnTargetsVive.depth = 3`
Number of possible depth spawn positions in the interaction area. Distance between spawn points defined by SPAWN DISTANCE parameter.

Head `GameObject SpawnTargetsVive.Head`
GameObject of the HMD prefab of the VST HMD.

height `int SpawnTargetsVive.height = 3`
Number of possible height spawn positions in the interaction area. Distance between spawn points defined by SPAWN DISTANCE parameter.

logDataFilename `string SpawnTargetsVive.logDataFilename = "data"`
Filename where the recorded data will be saved. An increasing index will be added at the end, e.g. writing "test_" here will generate "test_0.txt", "test_1.txt" and so on

trialsNumber `int SpawnTargetsVive.trialsNumber = 0`
Number of displayed stimuli. In our study we used 27, but can be changed.

spawnDistance `float SpawnTargetsVive.spawnDistance = 5`
Minimum distance between spawn positions (centimeters).

target `GameObject SpawnTargetsVive.target`
Prefab of the displayed stimuli (in our case, a circle).

targetSize `float SpawnTargetsVive.targetSize = 1`
Scale of the prefab used to display the stimuli. Will be applied as invariant scale factor.

userDistance `float SpawnTargetsVive.userDistance = 50`
Distance of the volume from the user (centimeters). Measured from the centre of the volume.

Volume `GameObject SpawnTargetsVive.Volume`
Root gameObject used as bounding volume for the stimuli grid: all the stimuli prefabs will be instantiated as children of this gameObject. Create an empty GameObject "Volume" and position it as needed in the scene.

width `int SpawnTargetsVive.width = 3`
Number of possible width spawn positions in the interaction area. Distance between spawn points defined by SPAWN DISTANCE parameter.

C.20 SwitchProjection Class Reference

Can be used to switch back and forth (by pressing the space bar on the keyboard) between a chosen camera pose and the one obtained through tracking (for debugging purposes).
Inherits MonoBehaviour.

Public Attributes

- bool enableCorrectedProjection
- float **t1** = 0.015f
- float **r1** = 62.5f

C.20.1 Member Data Documentation

enableCorrectedProjection `bool SwitchProjection.enableCorrectedProjection`
Set to true to use custom projection by default.

C.21 syncKinect Class Reference

Synchronizes the position of a chosen gameObject with the 3D world position of the point tracked by the Kinect, which is received from the socketReceiver class.
Inherits MonoBehaviour.

C.21.1 Detailed Description

Attach both syncKinect.cs and socketReceiver.cs to any gameObject, and then create an empty gameObject with tag 'kinect'. The position of the 'kinect' tagged gameObject will be updated with the 3D position of the tracked point, expressed with respect to the Kinect Reference Frame. To express the tracked point position with respect to the HTC Vive reference frame, the position of the Kinect in the HTC Vive reference frame must be known, and the tracked object must be set as child of a gameObject with the Kinect position (in the HTC Vive reference frame) as transform. To obtain such position it is sufficient to attach a Vive Tracker (or a controller) rigidly and in a known position with respect to the Kinect. To do this, we attached the Kinect to a perforated metal angular profile, and then attached a Vive Tracker on a 3D printer part which can house a ZED mini in a known position. We then obtained the transformation between the Zed Mini and the Kinect Camera through camera calibration, and since the transformation between the ZED mini and the Vive Tracker is known from the CAD of the 3D printed part, we can thus track the Kinect position (and therefore the 3D point tracked by the Kinect) in the HTC Vive Reference frame.

For more details on the process refer to the related chapter (Kinect Registration Module, section 3.0.3).

C.22 Target_tracker_pose Class Reference

Updates the target gameObject position, depending on the pose of the Vive Tracker used during the SPAAM procedure. The same tracker has been later used to track the hand and/or the Kinect position in the active task experimental session.

Inherits MonoBehaviour.

C.23 tinyCalibration Class Reference

Handles the checkerboard alignment procedure performed by each user before using the OST HMD. The approximate correct parameters must be obtained from the SPAAM procedure first. Starts the experimental sessions immediately after the alignment task.

Inherits MonoBehaviour.

Public Attributes

- float adjustmentShift = 0
- float adjustmentAngle = 0
- float focalStep = 0
- bool skipCalibration = false
- bool secondExperiment = false
- GameObject CheckerBoard

C.23.1 Detailed Description

The alignment procedure can be in two different states: Translation editing mode or Rotation editing mode. By default, the first active state is the Translation editing mode. The experimenter can align the virtual checkerboard with the instructions of the user depending on the perceived position of the virtual checkerboard with respect to the real one. The assigned key-bindings for the Translation editing mode are the following ones:

- Y: Switch to Rotation mode.
- W: Moves the virtual checkerboard forward in the Z axis (away from the user).
- A: Moves the virtual checkerboard backward in the X axis (to the left of the user).
- S: Moves the virtual checkerboard backwards in the Z axis (towards the user).
- D: Moves the virtual checkerboard forward in the X axis (to the right of the user).
- R: Moves the virtual checkerboard forward in the Y axis (towards the ceiling).
- F: Moves the virtual checkerboard backwards in the Y axis (towards the floor).
- X: Increases the adjustment step (each keypress results in a bigger translation/rotation).
- Z: Decreases the adjustment step (each keypress results in a smaller translation/rotation).
- Space: Saves the current parameters and continues to the experiment.

The assigned keybindings for the Rotation editing mode are the following ones:

- U: Switch to Translation mode.
- W: Increases the virtual checkerboard rotation over the X axis.
- A: Decreases the virtual checkerboard rotation over the Y axis.
- S: Decreases the virtual checkerboard rotation over the X axis.
- D: Increases the virtual checkerboard rotation over the Y axis.
- Q: Increases the virtual checkerboard rotation over the Z axis (clockwise with respect to the user optical axis).
- E: Decreases the virtual checkerboard rotation over the Z axis (anticlockwise with respect to the user optical axis).
- X: Increases the adjustment step (each keypress results in a bigger translation/rotation).
- Z: Decreases the adjustment step (each keypress results in a smaller translation/rotation).
- Space: Saves the current parameters and continues to the experiment.

C.23.2 Member Data Documentation

adjustmentAngle `float tinyCalibration.adjustmentAngle = 0`

Defines the step angle in used for each rotation of the checkerboard during the alignment.

adjustmentShift `float tinyCalibration.adjustmentShift = 0`
Defines the step size in mm used for each translation of the checkerboard during the alignment.

CheckerBoard `GameObject tinyCalibration.CheckerBoard`
GameObject with checkerboard_tracker_pose attached, which tracks the 3D world position of the checkerboard.

focalStep `float tinyCalibration.focalStep = 0`
Defines the step used to modify the scale factor.

secondExperiment `bool tinyCalibration.secondExperiment = false`
Set to true if performing the active alignment task.

skipCalibration `bool tinyCalibration.skipCalibration = false`
Set to true to skip the alignment procedure, and read an existing calibration file.

C.24 tinyCalibrationVive Class Reference

Handles the checkerboard alignment procedure performed by each user before using the VST HMD. The approximate correct parameters must be obtained from the SPAAM procedure first. Starts the experimental sessions immediately after the alignment task.
Inherits MonoBehaviour.

Public Attributes

- float adjustmentShift = 0
- float adjustmentAngle = 0
- float focalStep = 0
- bool skipCalibration = false
- bool secondExperiment = false
- GameObject CheckerBoard

C.24.1 Detailed Description

The alignment procedure can be in two different states: Translation editing mode or Rotation editing mode. By default, the first active state is the Translation editing mode. The experimenter can align the virtual checkerboard with the instructions of the user depending on the perceived position of the virtual checkerboard with respect to the real one. The assigned keybindings for the Translation editing mode are the following ones:

- Y: Switch to Rotation mode.

- W: Moves the virtual checkerboard forward in the Z axis (away from the user).
- A: Moves the virtual checkerboard backward in the X axis (to the left of the user).
- S: Moves the virtual checkerboard backwards in the Z axis (towards the user).
- D: Moves the virtual checkerboard forward in the X axis (to the right of the user).
- R: Moves the virtual checkerboard forward in the Y axis (towards the ceiling).
- F: Moves the virtual checkerboard backwards in the Y axis (towards the floor).
- X: Increases the adjustment step (each keypress results in a bigger translation/rotation).
- Z: Decreases the adjustment step (each keypress results in a smaller translation/rotation).
- Space: Saves the current parameters and continues to the experiment.

The assigned keybindings for the Rotation editing mode are the following ones:

- U: Switch to Translation mode.
- W: Increases the virtual checkerboard rotation over the X axis.
- A: Decreases the virtual checkerboard rotation over the Y axis.
- S: Decreases the virtual checkerboard rotation over the X axis.
- D: Increases the virtual checkerboard rotation over the Y axis.
- Q: Increases the virtual checkerboard rotation over the Z axis (clockwise with respect to the user optical axis).
- E: Decreases the virtual checkerboard rotation over the Z axis (anticlockwise with respect to the user optical axis).
- X: Increases the adjustment step (each keypress results in a bigger translation/rotation).
- Z: Decreases the adjustment step (each keypress results in a smaller translation/rotation).
- Space: Saves the current parameters and continues to the experiment.

C.24.2 Member Data Documentation

adjustmentAngle `float tinyCalibrationVive.adjustmentAngle = 0`

Defines the step angle in used for each rotation of the checkerboard during the alignment.

adjustmentShift `float tinyCalibrationVive.adjustmentShift = 0`

Defines the step size in mm used for each translation of the checkerboard during the alignment.

CheckerBoard `GameObject tinyCalibrationVive.CheckerBoard`
GameObject with checkerboard_tracker_pose attached, which tracks the 3D world position of the checkerboard.

focalStep `float tinyCalibrationVive.focalStep = 0`
Defines the step used to modify the scale factor.

secondExperiment `bool tinyCalibrationVive.secondExperiment =false`
Set to true if performing the active alignment task.

skipCalibration `bool tinyCalibrationVive.skipCalibration = false`
Set to true to skip the alignment procedure, and read an existing calibration file.

Bibliography

[1] https://unlicense.org/.

[2] Robert Allison. 'Analysis of the influence of vertical disparities arising in toed-in stereoscopic cameras'. In: *Journal of Imaging Science and Technology* 51.4 (2007), pp. 317–327.

[3] *Augmented Reality for Enterprise Alliance*. 2018 (accessed October 1, 2018).

[4] Magnus Axholt, Martin Skoglund, Stephen D Peterson, Matthew D Cooper, Thomas B Schön, Fredrik Gustafsson, Anders Ynnerman and Stephen R Ellis. 'Optical see-through head mounted display direct linear transformation calibration robustness in the presence of user alignment noise'. In: *Proceedings of the human factors and ergonomics society annual meeting*. Vol. 54. 28. SAGE Publications Sage CA: Los Angeles, CA. 2010, pp. 2427–2431.

[5] Mahdi Azmandian, Mark Hancock, Hrvoje Benko, Eyal Ofek and Andrew D Wilson. 'Haptic retargeting: Dynamic repurposing of passive haptics for enhanced virtual reality experiences'. In: *Proceedings of the 2016 chi conference on human factors in computing systems*. 2016, pp. 1968–1979.

[6] Giorgio Ballestin. *An AR Registration Framework for Video See-Through and Optical See-Through Head Mounted Displays*. https://github.com/3632741/AR-Registration-Framework-PhD-Thesis. 2021.

[7] Giorgio Ballestin, Manuela Chessa and Fabio Solari. 'Assessment of Optical See-Through Head Mounted Display Calibration for Interactive Augmented Reality'. In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*. 2019.

[8] Giorgio Ballestin, Manuela Chessa and Fabio Solari. 'A Registration Framework for the Comparison of Video and Optical See-Through Devices in Interactive Augmented Reality'. In: *IEEE Access* 9 (2021), pp. 64828–64843. DOI: 10.1109/ACCESS.2021.3075780.

[9] Giorgio Ballestin, Fabio Solari and Manuela Chessa. 'Perception and Action in Peripersonal Space: a comparison between Video and Optical see-through Augmented Reality Devices'. In: *Adjunct Proceedings of the IEEE International Symposium for Mixed and Augmented Reality 2018 (To appear)*. 2018.

[10] Giorgio Ballestin, Fabio Solari and Manuela Chessa. 'Perception and action in peripersonal space: A comparison between video and optical see-through augmented reality devices'. In: *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE. 2018, pp. 184–189.

[11] Chiara Bassano, Fabio Solari and Manuela Chessa. 'Studying Natural Human-computer Interaction in Immersive Virtual Reality: A Comparison between Actions in the Peripersonal and in the Near-action Space.' In: *VISIGRAPP (2: HUCAPP)*. 2018, pp. 108–115.

[12] Adrian Borrego, Jorge Latorre, Mariano Alcañiz and Roberto Llorens. 'Comparison of Oculus Rift and HTC Vive: feasibility for virtual reality-based exploration, navigation, exergaming, and rehabilitation'. In: *Games for health journal* 7.3 (2018), pp. 151–156.

[13] Sébastien Bottecchia, Jean-Marc Cieutat, Christophe Merlo and Jean-Pierre Jessel. 'A new AR interaction paradigm for collaborative teleassistance system: the POA'. In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 3.1 (2009), pp. 35–40.

[14] Lisa Gottesfeld Brown. 'A survey of image registration techniques'. In: *ACM computing surveys (CSUR)* 24.4 (1992), pp. 325–376.

[15] Emanuele Maria Calabrò, Fabrizio Cutolo, Marina Carbone and Vincenzo Ferrari. 'Wearable augmented reality optical see through displays based on integral imaging'. In: *International Conference on Wireless Mobile Communication and Healthcare*. Springer. 2016, pp. 345–356.

[16] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei and Y. A. Sheikh. 'OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[17] Zhe Cao, Tomas Simon, Shih-En Wei and Yaser Sheikh. 'Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields'. In: *CVPR*. 2017.

[18] Stuart K Card. *The psychology of human-computer interaction*. Crc Press, 2018.

[19] Nadia Cattari, Fabrizio Cutolo, Renzo D'amato, Umberto Fontana and Vincenzo Ferrari. 'Toed-in vs parallel displays in video see-through head-mounted displays for close-up view'. In: *IEEE Access* 7 (2019), pp. 159698–159711.

[20] Manuela Chessa and Fabio Solari. '[POSTER] Walking in Augmented Reality: An Experimental Evaluation by Playing with a Virtual Hopscotch'. In: *Mixed and Augmented Reality (ISMAR-Adjunct), 2017 IEEE International Symposium on*. IEEE. 2017, pp. 143–148.

[21] Manuela Chessa and Fabio Solari. 'A geometric model of spatial distortions in virtual and augmented environments'. In: *2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS)*. IEEE. 2018, pp. 55–60.

[22] Fabrizio Cutolo, Umberto Fontana and Vincenzo Ferrari. 'Perspective preserving solution for quasi-orthoscopic video see-through HMDs'. In: *Technologies* 6.1 (2018), p. 9.

[23] James E Cutting. 'How the eye measures reality and virtual reality'. In: *Behavior Research Methods, Instruments, & Computers* 29.1 (1997), pp. 27–36.

[24] Saverio Debernardis, Michele Fiorentino, Michele Gattullo, Giuseppe Monno and Antonio Emmanuele Uva. 'Text readability in head-worn displays: Color and style optimization in video versus optical see-through devices'. In: *IEEE transactions on visualization and computer graphics* 20.1 (2013), pp. 125–139.

[25] Catherine Diaz, Michael Walker, Danielle Albers Szafir and Daniel Szafir. 'Designing for depth perceptions in augmented reality'. In: *2017 IEEE international symposium on mixed and augmented reality (ISMAR)*. IEEE. 2017, pp. 111–122.

[26] Neil A Dodgson. 'Variation and extrema of human interpupillary distance'. In: *Stereoscopic Displays and Virtual Reality Systems XI*. Vol. 5291. International Society for Optics and Photonics. 2004, pp. 36–47.

[27] Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT press, 1993.

[28] Martin A Fischler and Robert C Bolles. 'Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography'. In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[29] James D Foley, Foley Dan Van, Andries Van Dam, Steven K Feiner, John F Hughes, Edward Angel and J Hughes. *Computer graphics: principles and practice*. Vol. 12110. Addison-Wesley Professional, 1996.

[30] Forrester. *Forrester 2017 predictions*. 2017 (accessed October 1, 2018).

[31] Henry Fuchs, Mark A Livingston, Ramesh Raskar, Kurtis Keller, Jessica R Crawford, Paul Rademacher, Samuel H Drake, Anthony A Meyer et al. 'Augmented reality visualization for laparoscopic surgery'. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 1998, pp. 934–943.

[32] Vittorio Gallese, Luciano Fadiga, Leonardo Fogassi and Giacomo Rizzolatti. 'Action recognition in the premotor cortex'. In: *Brain* 119.2 (1996), pp. 593–609.

[33] Yakup Genc, Frank Sauer, Fabian Wenzel, Mihran Tuceryan and Nassir Navab. 'Optical see-through HMD calibration: A stereo method validated with a video see-through system'. In: *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*. IEEE. 2000, pp. 165–174.

[34] Yakup Genc, Mihran Tuceryan, Ali Khamene and Nassir Navab. 'Optical see-through calibration with vision-based trackers: Propagation of projection matrices'. In: *Proceedings IEEE and ACM International Symposium on Augmented Reality*. IEEE. 2001, pp. 147–156.

[35] Yakup Genc, Mihran Tuceryan and Nassir Navab. 'Practical solutions for calibration of optical see-through devices'. In: *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*. IEEE Computer Society. 2002, p. 169.

[36] Timofey Y Grechkin, Tien Dat Nguyen, Jodie M Plumert, James F Cremer and Joseph K Kearney. 'How does presentation method and measurement protocol affect distance estimation in real and virtual environments?' In: *ACM Transactions on Applied Perception (TAP)* 7.4 (2010), p. 26.

[37] Jens Grubert, Yuta Itoh, Kenneth Moser and J Edward Swan. 'A survey of calibration methods for optical see-through head-mounted displays'. In: *IEEE transactions on visualization and computer graphics* 24.9 (2018), pp. 2649–2662.

[38] Jens Grubert, Johannes Tuemle, Ruediger Mecke and Michael Schenk. 'Comparative User Study of two See-through Calibration Methods.' In: *VR* 10 (2010), pp. 269–270.

[39] Sandra G Hart and Lowell E Staveland. 'Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research'. In: *Advances in psychology*. Vol. 52. Elsevier, 1988, pp. 139–183.

[40] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[41] Nicolas S Holliman, Neil A Dodgson, Gregg E Favalora and Lachlan Pockett. 'Three-dimensional displays: a review and applications analysis'. In: *IEEE transactions on Broadcasting* 57.2 (2011), pp. 362–371.

[42] Ian P Howard and Brian J Rogers. 'Seeing in depth, volume 2: Depth perception'. In: *Ontario, Canada: I. Porteous* (2002).

[43] Fu-Chung Huang, David P Luebke and Gordon Wetzstein. 'The light field stereoscope.' In: *SIGGRAPH Emerging Technologies*. 2015, pp. 24–1.

[44] William James. *The principles of psychology*. Vol. 1. Cosimo, Inc., 2007.

[45] Marc Jeannerod et al. 'The representing brain: Neural correlates of motor intention and imagery'. In: *Behavioral and Brain sciences* 17.2 (1994), pp. 187–201.

[46] J Adam Jones, J Edward Swan, Gurjot Singh, Eric Kolstad and Stephen R Ellis. 'The effects of virtual reality, augmented reality, and motion parallax on egocentric depth perception'. In: *Proceedings of the 5th symposium on Applied perception in graphics and visualization*. 2008, pp. 9–14.

[47] Adolf Jost. *Die Assoziationsfestigkeit in ihrer Abhängigkeit von der Verteilung der Wiederholungen*. L. Voss, 1897.

[48] Ricardo Jota, Albert Ng, Paul Dietz and Daniel Wigdor. 'How fast is fast enough? a study of the effects of latency in direct-touch pointing tasks'. In: *Proceedings of the sigchi conference on human factors in computing systems*. 2013, pp. 2291–2300.

[49] M Carmen Juan and Jérôme Calatrava. 'An augmented reality system for the treatment of phobia to small animals viewed via an optical see-through HMD: Comparison with a similar system viewed via a video see-through HMD'. In: *International Journal of Human-Computer Interaction* 27.5 (2011), pp. 436–449.

[50] Daniel Kahneman and Amos Tversky. 'Prospect theory: An analysis of decision under risk'. In: *Handbook of the fundamentals of financial decision making: Part I*. World Scientific, 2013, pp. 99–127.

[51] Hirokazu Kato and Mark 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)*. IEEE. 1999, pp. 85–94.

[52] Robert S Kennedy, Norman E Lane, Kevin S Berbaum and Michael G Lilienthal. 'Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness'. In: *The international journal of aviation psychology* 3.3 (1993), pp. 203–220.

[53] Mathias Klinghammer, Immo Schütz, Gunnar Blohm and Katja Fiehler. 'Allocentric information is used for memory-guided reaching in depth: A virtual reality study'. In: *Vision research* 129 (2016), pp. 13–24.

[54] Gregory Kramida. 'Resolving the vergence-accommodation conflict in head-mounted displays'. In: *IEEE transactions on visualization and computer graphics* 22.7 (2015), pp. 1912–1931.

[55] Ernst Kruijff, J Edward Swan and Steven Feiner. 'Perceptual issues in augmented reality revisited'. In: *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*. IEEE. 2010, pp. 3–12.

[56] Sangyoon Lee and Hong Hua. 'A robust camera-based method for optical distortion calibration of head-mounted displays'. In: *Journal of Display Technology* 11.10 (2014), pp. 845–853.

[57] Seok-Han Lee, Sang-Keun Lee and Jong-Soo Choi. 'Correction of radial distortion using a planar checkerboard pattern and its image'. In: *IEEE Transactions on Consumer Electronics* 55.1 (2009), pp. 27–33.

[58] Patrick Maier, Arindam Dey, Christian AL Waechter, Christian Sandor, Marcus Tönnis and Gudrun Klinker. 'An empiric evaluation of confirmation methods for optical see-through head-mounted display calibration'. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE. 2011, pp. 267–268.

[59] Erin McGarrity, Yakup Genc, Mihran Tuceryan, Charles Owen and Nassir Navab. 'A new system for online quantitative evaluation of optical see-through augmentation'. In: *Proceedings IEEE and ACM International Symposium on Augmented Reality*. IEEE. 2001, pp. 157–166.

[60] Paul Milgram, Haruo Takemura, Akira Utsumi and Fumio Kishino. 'Augmented reality: A class of displays on the reality-virtuality continuum'. In: *Telemanipulator and telepresence technologies*. Vol. 2351. International Society for Optics and Photonics. 1995, pp. 282–292.

[61] George A Miller. 'The magical number seven, plus or minus two: Some limits on our capacity for processing information.' In: *Psychological review* 63.2 (1956), p. 81.

[62] Mark Mon-Williams and James R Tresilian. 'Ordinal depth information from accommodation?' In: *Ergonomics* 43.3 (2000), pp. 391–404.

[63] Kenneth Moser, Yuta Itoh, Kohei Oshima, J Edward Swan, Gudrun Klinker and Christian Sandor. 'Subjective evaluation of a semi-automatic optical see-through head-mounted display calibration technique'. In: *IEEE transactions on visualization and computer graphics* 21.4 (2015), pp. 491–500.

[64] Mahdi Nabiyouni, Siroberto Scerbo, Doug A Bowman and Tobias Höllerer. 'Relative effects of real-world and Virtual-World latency on an augmented reality Training Task: an ar simulation experiment'. In: *Frontiers in ICT* 3 (2017), p. 34.

[65] Abdeldjallil Naceri, Ryad Chellali and Thierry Hoinville. 'Depth perception within peripersonal space using head-mounted display'. In: *Presence: Teleoperators and Virtual Environments* 20.3 (2011), pp. 254–272.

[66] S Nagata. 'How to reinforce perception of depth in single two-dimensional pictures'. In: *Spatial displays and spatial instruments* 29 (1987).

[67] Phillip E Napieralski, Bliss M Altenhoff, Jeffrey W Bertrand, Lindsay O Long, Sabarish V Babu, Christopher C Pagano, Justin Kern and Timothy A Davis. 'Near-field distance perception in real and virtual environments using both verbal and action responses'. In: *ACM Transactions on Applied Perception (TAP)* 8.3 (2011), p. 18.

[68] Nassir Navab, Siavash Zokai, Yakup Genc and Enylton Machado Coelho. 'An on-line evaluation system for optical see-through augmented reality'. In: *IEEE Virtual Reality 2004*. IEEE. 2004, pp. 245–246.

[69] Diederick C Niehorster, Li Li and Markus Lappe. 'The accuracy and precision of position and orientation tracking in the HTC Vive virtual reality system for scientific research'. In: *i-Perception* 8.3 (2017), p. 2041669517708205.

[70] Mika E Ono, Josée Rivest and Hiroshi Ono. 'Depth perception as a function of motion parallax and absolute-distance information.' In: *Journal of Experimental Psychology: Human Perception and Performance* 12.3 (1986), p. 331.

[71] Charles B Owen, Ji Zhou, Arthur Tang and Fan Xiao. 'Display-relative calibration for optical see-through head-mounted displays'. In: *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society. 2004, pp. 70–78.

[72] B Prescott and GF McLean. 'Line-based correction of radial lens distortion'. In: *Graphical Models and Image Processing* 59.1 (1997), pp. 39–47.

[73] Fred H Previc. 'The neuropsychology of 3-D space.' In: *Psychological bulletin* 124.2 (1998), p. 123.

[74] Wolfgang Prinz. 'Ideo-motor action'. In: *Perspectives on perception and action* (1987), pp. 47–76.

[75] Holger Regenbrecht and Thomas Schubert. 'Measuring presence in augmented reality environments: design and a first test of a questionnaire'. In: *Porto, Portugal* (2002).

[76] Holger Regenbrecht and Thomas Schubert. 'Real and illusory interactions enhance presence in virtual environments'. In: *Presence: Teleoperators & Virtual Environments* 11.4 (2002), pp. 425–434.

[77] Rebekka S Renner, Boris M Velichkovsky and Jens R Helmert. 'The perception of egocentric distances in virtual environments-a review'. In: *ACM Computing Surveys (CSUR)* 46.2 (2013), p. 23.

[78] Giacomo Rizzolatti, Luciano Fadiga, Vittorio Gallese and Leonardo Fogassi. 'Premotor cortex and the recognition of motor actions'. In: *Cognitive brain research* 3.2 (1996), pp. 131–141.

[79] Jannick P Rolland and Henry Fuchs. 'Optical versus video see-through head-mounted displays in medical visualization'. In: *Presence: Teleoperators & Virtual Environments* 9.3 (2000), pp. 287–309.

[80] Jannick P Rolland, Richard L Holloway and Henry Fuchs. 'Comparison of optical and video see-through, head-mounted displays'. In: *Telemanipulator and Telepresence Technologies*. Vol. 2351. International Society for Optics and Photonics. 1995, pp. 293–307.

[81] Carlos Salas Rosales, Grant Pointon, Haley Adams, Jeanine Stefanucci, Sarah Creem-Regehr, William B Thompson and Bobby Bodenheimer. 'Distance judgments to on-and off-ground objects in augmented reality'. In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2019, pp. 237–243.

[82] Thomas Schubert, Frank Friedmann and Holger Regenbrecht. 'The experience of presence: Factor analytic insights'. In: *Presence: Teleoperators & Virtual Environments* 10.3 (2001), pp. 266–281.

[83] Volkan Sevinc and Mehmet Ilker Berkman. 'Psychometric evaluation of Simulator Sickness Questionnaire and its variants as a measure of cybersickness in consumer virtual environments'. In: *Applied Ergonomics* 82 (2020), p. 102958.

[84] Tomas Simon, Hanbyul Joo, Iain Matthews and Yaser Sheikh. 'Hand Keypoint Detection in Single Images using Multiview Bootstrapping'. In: *CVPR*. 2017.

[85] Gurjot Singh, J Edward Swan II, J Adam Jones and Stephen R Ellis. 'Depth judgment measures and occluding surfaces in near-field augmented reality'. In: *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization*. ACM. 2010, pp. 149–156.

[86] Andrei State, Kurtis P Keller and Henry Fuchs. 'Simulation-based design and rapid prototyping of a parallax-free, orthoscopic video see-through head-mounted display'. In: *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05)*. IEEE. 2005, pp. 28–31.

[87] J Edward Swan, Adam Jones, Eric Kolstad, Mark A Livingston and Harvey S Smallman. 'Egocentric depth judgments in optical, see-through augmented reality'. In: *IEEE transactions on visualization and computer graphics* 13.3 (2007), pp. 429–442.

[88] J Edward Swan, Liisa Kuparinen, Scott Rapson and Christian Sandor. 'Visually perceived distance judgments: Tablet-based augmented reality versus the real world'. In: *International Journal of Human–Computer Interaction* 33.7 (2017), pp. 576–591.

[89] J Edward Swan, Mark A Livingston, Harvey S Smallman, Dennis Brown, Yohan Baillot, Joseph L Gabbard and Deborah Hix. 'A perceptual matching technique for depth judgments in optical, see-through augmented reality'. In: *IEEE Virtual Reality Conference (VR 2006)*. IEEE. 2006, pp. 19–26.

[90] Akinari Takagi, Shoichi Yamazaki, Yoshihiro Saito and Naosato Taniguchi. 'Development of a stereo video see-through HMD for AR systems'. In: *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*. IEEE. 2000, pp. 68–77.

[91] Arthur Tang, Ji Zhou and Charles Owen. 'Evaluation of calibration procedures for optical see-through head-mounted displays'. In: *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society. 2003, p. 161.

[92] Makoto Tomioka, Sei Ikeda and Kosuke Sato. 'Approximated user-perspective rendering in tablet-based augmented reality'. In: *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE. 2013, pp. 21–28.

[93] James R Tresilian and Mark Mon-Williams. 'A curious illusion suggests complex cue interactions in distance perception.' In: *Journal of Experimental Psychology: Human Perception and Performance* 25.3 (1999), p. 677.

[94] Emanuele Trucco and Alessandro Verri. *Introductory techniques for 3-D computer vision*. Vol. 201. Prentice Hall Englewood Cliffs, 1998.

[95] Roger Tsai. 'A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses'. In: *IEEE Journal on Robotics and Automation* 3.4 (1987), pp. 323–344.

[96] Mihran Tuceryan, Yakup Genc and Nassir Navab. 'Single-point active alignment method (SPAAM) for optical see-through HMD calibration for augmented reality'. In: *Presence: Teleoperators & Virtual Environments* 11.3 (2002), pp. 259–276.

[97] Mihran Tuceryan, Douglas S. Greer, Ross T. Whitaker, David E. Breen, Chris Crampton, Eric Rose and Klaus H Ahlers. 'Calibration requirements and procedures for a monitor-based augmented reality system'. In: *IEEE Transactions on Visualization and Computer Graphics* 1.3 (1995), pp. 255–273.

[98] Ivan Valentini, Giorgio Ballestin, Chiara Bassano, Fabio Solari and Manuela Chessa. 'Improving Obstacle Awareness to Enhance Interaction in Virtual Reality'. In: *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2020, pp. 44–52.

[99] Cyril Vienne, Justin Plantier, Pascaline Neveu and Anne-Emmanuelle Priot. 'The role of vertical disparity in distance and depth perception as revealed by different stereo-camera configurations'. In: *i-Perception* 7.6 (2016), p. 2041669516681308.

[100] Alain Viguier, Gilles Clement and Yves Trotter. 'Distance perception within near visual space'. In: *Perception* 30.1 (2001), pp. 115–124.

[101] Ruixuan Walter Hannah andLi, Justin Munafo, Christopher Curry, Nicolette Peterson and Thomas. Stoffregen. *A brief explanation of the Simulator Sickness Questionnaire (SSQ)*. `https://doi.org/10.13020/XAMG-CS69`.

[102] Rebecca AT Weast and Dennis R Proffitt. 'Can I reach that? Blind reaching as an accurate measure of estimated reachable distance'. In: *Consciousness and cognition* 64 (2018), pp. 121–134.

[103] Shih-En Wei, Varun Ramakrishna, Takeo Kanade and Yaser Sheikh. 'Convolutional pose machines'. In: *CVPR*. 2016.

[104] Hsin-Kai Wu, Silvia Wen-Yu Lee, Hsin-Yi Chang and Jyh-Chong Liang. 'Current status, opportunities and challenges of augmented reality in education'. In: *Computers & education* 62 (2013), pp. 41–49.

[105] Gang Xu and Zhengyou Zhang. *Epipolar geometry in stereo, motion and object recognition: a unified approach*. Vol. 6. Springer Science & Business Media, 2013.

[106] Z. Zhang. 'A flexible new technique for camera calibration'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Genova, Italy
May 2021

Giorgio Ballestin