TU WIEN Informatics

Hand Tracking in Colocated Multi-User Virtual Reality

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dennis Reimer, M.Sc.
Registration Number 11728991

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Hannes Kaufmann
Second advisor: Prof. Dr. Daniel Scherzer

The dissertation has been reviewed by:

Tiare Feuchtner

Eike Langbehn

Vienna, 1st July, 2024

Dennis Reimer

Erklärung zur Verfassung der Arbeit

Dennis Reimer, M.Sc.

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

Wien, 1. Juli 2024

Dennis Reimer

Acknowledgements

A dissertation like this is certainly not the result of a single person, which is why there are many people whom I would like to thank and who have made it possible for me to produce this thesis in this form. First of all, I would like to thank my supervisor Hannes Kaufmann, who always supported me with helpful feedback, guidance, valuable advice, and a lot of patience. I would also like to thank my second supervisor Daniel Scherzer, who made the initial start to my PhD possible and also who has always helped me with advice and support.

I also thank Iana Podkosova for her help in finding the topic, for the very interesting exchange, and for her help as coauthor on two articles. Of course, I would also like to thank the entire VR Group at Vienna University of Technology, which welcomed me with open arms and offered me a valuable exchange of ideas.

Many thanks also go to Tiare Feuchtner and Eike Langbehn, who kindly agreed to review the thesis.

This thesis was funded by Ravensburg-Weingarten University, who kindly employed me for the research and writing of my dissertation, and the TU Wien Bibliothek for financial support through its Open Access Funding Program.

I would especially like to thank my fiancée Anna-Catrin. Her moral support and the fact that she always had an open ear helped me through the difficult phases, and this work would never have been possible without her. To my daughter Linnea, who makes every day a beautiful one for me and who I hope I can inspire to always be curious and explore the world.

Many thanks go to my parents Peter and Renate Reimer, without whom my path to my studies, my enthusiasm for computer science and ultimately this thesis would not have been possible. Additional thanks go to my good friend Thorsten, who was always open to an exchange and helped me out as a test person in every experiment.

Finally, thanks go to all the testers who agreed to participate in the user tests that were part of this thesis and thus helped to generate valuable results.

Abstract

To enhance immersion in virtual reality applications, this dissertation addresses the challenges and opportunities of implementing natural hand interactions through hardware-based hand tracking, particularly in the context of multi-user colocated VR environments. The research introduces 'EasyHand', a hand-tracking framework that unifies detection mapping, visualization, interaction, and networking for several tracking systems to facilitate intuitive hand tracking for visualization and interaction while addressing the limitations of tracking range dead spots in colocated scenarios.

The first experiment describes a novel approach for creating colocated multi-user VR scenarios for SLAM-tracked (Simultaneous Localization and Mapping) VR headsets without the need for external tracking cameras to continuously track all users. By leveraging the hand recognition capabilities of the VR headset, the system synchronizes the virtual space for colocated users. This method is compared to alternative approaches such as initial positioning and ArUco marker recognition, with a comprehensive evaluation of accuracy, consistency, and simplicity, demonstrating the superior performance of the proposed hand tracking-based calibration method.

The dissertation proceeds with an experiment that demonstrates a method for transforming hand data detected via an RGB camera and the MediaPipe framework into 3D space. This technique includes user-specific hand length estimation to determine 3D hand positions, enabling the detection of multiple hands simultaneously. This allows a hand detection system to track the hands of other users and thus also support their detection systems in the event that they are not able to see these hands. Comparative analysis with Oculus Quest and Leap Motion, conducted under different conditions (static & dynamic) and distances from the tracking device, confirms the effectiveness of the proposed method, with significantly extended tracking ranges for colocated scenarios.

Finally, the dissertation explores methods for assigning tracked hands to colocated virtual users, introducing an algorithm that leverages past assignments to enhance future assignments' robustness and effectiveness. Multiple assignment algorithms are evaluated, highlighting the precision of the proposed algorithm.

Overall, this work provides initial insights into calibrating multi-user environments, compensating for tracking loss and assignment of hands to users for colocated VR scenarios, while maintaining user-specific interactions, representing a substantial advancement in VR technology.

Contents

CHAPTER 1

Introduction

"Curiosity is the most powerful driving force in the universe because it can overcome the two greatest braking forces in the universe: Reason and Fear."
— Walter Moers - The City Of Dreaming Books

Hand tracking has emerged as a popular feature in Virtual Reality (VR) systems, offering users a multitude of interaction possibilities while granting maximum hand freedom without the need for handheld controllers. Controllers are the modern standard method for interaction, supporting direct manipulation of virtual content via a series of buttons and gestures through stable tracking of hand position as well as multimodal feedback (vibration and sound, passive haptic feedback from buttons, triggers, etc.). However, since the use of controllers obscures the user's own hands and makes free gesticulation and the use of props impossible, hand tracking and hand interactions offer significantly more scope and more natural interaction possibilities. While the majority of applications and use cases for virtual reality in general and hand interactions in particular have historically focused on single-user experiences, there is a growing trend towards multi-user approaches. Specifically, colocated multi-user environments, where individuals share both virtual and physical spaces, hold the promise of immersive virtual activities.

As there is a push in colocated VR due to affordability, technical feasibility and user acceptance, new challenges and opportunities for hand tracking also arise: hands of several users must be tracked accurately and assigned to the correct user. Also, tracking redundancy when multiple systems recognize the same hand, or when one system recognizes multiple hands, can be used to improve tracking stability and range, or even to synchronize the coordinate systems of the head-mounted displays (HMDs). Solving this challenge and utilizing the mentioned possibilities is the goal that this work tries to solve

This dissertation shows solutions that are designed to enable colocated VR environments for SLAM-tracked virtual reality headsets equipped with hand-tracking technology.

Furthermore, it explores the usability of this approach in comparison to more complex alternatives. Additionally, this work explores the limitations encountered when using hand tracking in such scenarios and propose strategies to overcome these limitations. These enable the integration of multiple hand-tracking systems that can assist each other in cases where hands become obscured.

These solutions are combined and integrated in a comprehensive system that provides an accessible framework for developers who want to create a unified application independently from the tracking system and are able to create colocated virtual environments solely reliant on hand-tracking technology and do not require additional hardware such as an external camera rig. Additionally, it leverages the transformation of two-dimensional tracked hands into the three-dimensional world through pose estimation and robust user assignment. Furthermore, this work evaluates the usability of these systems and explore potential enhancements that can improve usability and enable more effective interactions within colocated multi-user VR environments, ultimately supporting a more natural VR experience and eliminating the need for physical controllers.

1.1 Motivation

Since the release of Oculus Rift and HTC Vive, virtual reality (VR) has experienced significant growth within the commercial sector [34]. This expansion is evident both in personal gaming, with Sony's PSVR and the VR stores offered by HTC Vive and Oculus, as well as in the industrial sector, where VR is extensively used for training (e.g. Strivr[1]) and marketing purposes (e.g. IKEA Virtual Showroom[2]). VR has evolved to offer the most immersive digital experiences available and has become increasingly user-friendly and easy to set up, featuring fewer cables and cameras [2][34]. With the advent of Simultaneous Localization and Mapping (SLAM) technology, VR systems can now function without outside-in tracking technology, enhancing their portability.

The trend to minimize hardware dependency continues with the emerging technology in optical hand-tracking. This allows users to interact within the virtual world solely using their hands, eliminating the need for additional controllers or gloves. Therefore the virtual representation of the hand could better match the real hand and the interaction with bare hands in the virtual environment may have a positive influence on the sense or presence.

In recent years, a growing desire among people to share digital experiences was observed, as evidenced by the increasing usage of social media platforms [6]. This desire for shared experiences extends to VR, particularly in colocated environments. VR arenas and similar spaces (e.g. YULLBE[3]) take advantage of this technology to create collaborative and immersive virtual reality experiences.

[1]Strivr: `https://www.strivr.com/` (Accessed: 2024-06-21)
[2]IKEA VR Showroom: `https://present.digital/ikea/` (Accessed: 2024-06-21)
[3]YULLBE: `https://yullbe.com/` (Accessed: 2024-06-21)

Many of these colocated experiences currently rely on an array of additional hardware, including body-worn trackers, external camera setups, and cable-bound head-mounted displays attached to user-worn backpacks. The optimization of the setup effort and space limitation of such scenarios would benefit from the development of SLAM-tracked headsets capable of creating colocated environments without the need for extra hardware or cameras. Furthermore, the integration of hand-tracking capabilities could enhance the sense of presence and collaboration, especially in cases where a single user's hand tracking may fail due to obscured hands, where limited tracking volume and handling of physical objects may cause the hands to not be properly visible to the tracking sensor. In colocated setups, multiple hand-tracking systems could assist each other, enlarging the tracking range in which a hand can be detected. This would result in a scenario where users only need to wear a lightweight HMD to fully immerse themselves in a shared VR experience, making it independent of the surrounding hardware setup and having the devices support each other in their tracking. Ideally, this is then not limited to a specific HMD and thus allows interoperability, which to our current knowledge is not yet possible with current headsets (e.g. Meta Quest or Apple Vision Pro).

1.2 Problem Statement and Contribution

In this section we will briefly summarize what the goals of this work are, what problems we encounter and what this work offers as contributions to achieve these goals.

Overall, we want to advance what has been achieved so far in the area of multi-user to improve hand-based interactions within a physical and virtual shared space in a virtual reality (VR) application (colocated VR). Such a shared area offers the advantage that users often are within sight of other users. This means that their hand recognition systems can also see the hands of the other users. Using this fact to enable the systems to support each other in tracking is one of the overarching goals of this work. This makes it possible for the systems to recognize hands and make them virtually usable in situations in which hands may be hidden or not visible to their own system (for example, when they are held behind the back).

When developing the methods, it was also important for us to limit ourselves to hardware that is easily available to a potential end user without incurring high additional costs. This results in the use of widely available SLAM tracked headsets (such as the Meta Quest) and the use of RGB cameras in our work, as this allows the use of wide-spread webcams without the need for expensive hardware such as depth cameras. This should increase the range of users who can easily use the solution. Of course, these limitations also entail problems which, to our knowledge, have not yet been solved. These problems and our contribution to solving them are discussed below.

Numerous hardware options and implementations are available to enable hand tracking in virtual reality, such as the Leap Motion Sensor, the HTC hand tracking solution or Megatrack [43][14][37]. However, each of these implementations originates from different manufacturers and comes with its own software development kit (SDK). Consequently, each implementation possesses its unique detection events, gesture recognition, finger joint index mapping, and other specific features. This diversity makes it challenging to develop hand-tracking applications that work seamlessly across multiple VR systems. Developers are often forced to choose one particular system, such as Leap Motion, Meta Quest, or Vive, to focus on their development efforts. Thus, this thesis introduces **a comprehensive hand-tracking framework that consolidates multiple hand-tracking APIs**, simplifying the development process for applications across various systems. Furthermore, it **facilitates colocation capabilities for all supported systems** with the capability to send networking messages between all connected users (chapter 3). All the implementations presented in this thesis are conducted within this unified framework.

Many colocation implementations require additional sensors or cameras to track all users in a shared environment [122][15]. This introduces a more complex setup, which requires extra effort and expenses. Such a setup runs counter to the purpose of SLAM-tracked headsets, which are designed to be independent of external cameras for positional tracking. To address these issues, this thesis proposes **a method to enable colocation for SLAM-tracked headsets using only tracked hands** (chapter 4), thus synchronizing the self-contained coordinate systems of SLAM headsets and eliminating the need for external tracking systems.

To avoid ambiguity in tracking hands of multiple people, commercial hand tracking systems are typically limited to tracking only two hands; left and right [37]. However, in scenarios where tracking more hands is required, as when multiple users are physically colocated, alternative tracking methods must be considered. Ideally, additional costly hardware should be avoided (such as RGB-D sensors), restricting the choice to RGB-based tracking methods, such as the MediaPipe framework [64]. The challenge lies in the absence of 3D pose capabilities in such systems due to missing depth sensors. In response, this thesis proposes a **method to transform a two-dimensional tracked hand with relative depth data of finger joints into a three-dimensional representation. This work compares this approach with state-of-the-art systems and demonstrate its high accuracy, enabling interactions and significantly extending the tracking range, rendering it suitable for assistance in colocated multi-user scenarios** (chapter 5).

In colocated setups, where the limitation of only two concurrent hands is not given, it is possible to track more than two hands or multiple left or right hands simultaneously. In such cases, a system must assign each tracked hand to a virtual user. While straightforward in cases where users are far apart, this assignment becomes non-trivial in close proximity (within an arms length). This thesis presents **a solution for accurately assigning tracked hands to virtual users, achieving a remarkable 99% accuracy, whether in close or far proximity. Importantly, this method relies solely on positional data from hands and users gained from the HMD and does not require additional hardware** (chapter 6).

In summary, we aim to offer the following contributions:

- **I**: A framework that offers versatility for use in both single- and multi-user scenarios, with the ability to seamlessly switch between tracking systems.
- **II**: A method to calibrate colocated virtual environments for SLAM-tracked headsets using only tracked hands.
- **III**: A method to position a two-dimensionally tracked hand in a three-dimensional space.
- **IV**: Enabling the simultaneous tracking and virtual representation of more than two hands.
- **V**: A method to assign virtual hands to the corresponding user in colocated multi-user environments.

We aim to have a system that offers the creation of colocated VR environments, utilizing RGB hand tracking to monitor all visible hands in the view frustum and assign them to the respective users, even when their hands are obscured to their own tracking systems, potentially facilitating tracking and interaction with the virtual environment even when hands are obscured.

1.3 Resulting Publications

This PhD dissertation is the result of the following peer-reviewed publications, which are incorporated as published, with minor changes to conform to the overall thesis:

1. Dennis Reimer, Iana Podkosova, Daniel Scherzer and Hannes Kaufmann. "Colocation for SLAM-Tracked VR Headsets with Hand Tracking". In the journal of Advances in Seated Virtual Reality, Computers 2021, 10(5), 58, 2021.
2. Dennis Reimer, Iana Podkosova, Daniel Scherzer and Hannes Kaufmann. "Evaluation and improvement of HMD-based and RGB-based hand tracking solutions in VR". In the journal of Beyond Touch: Free Hand Interaction in Virtual Environments, Frontiers in Virtual Reality, 4:1169313, 2023.
3. Dennis Reimer, Daniel Scherzer and Hannes Kaufmann. "Ownership Estimation for Tracked Hands in a Colocated VR Environment". In the proceedings of the 33rd International Conference on Artificial Reality and Telexistence & Eurographics Symposium on Virtual Environments (ICAT-EGVE), Dublin, Ireland, pp. 105-114, 2023.

As the first author of the publications listed in this thesis, the main work was also done by me. This included the design, implementation, execution of the experiments, analysis, and writing of the publications. Other authors were involved in a supporting role and provided advice during the conception, writing, and analysis. This also applies to the 'EasyHand' framework presented, where the design and implementation of which was carried out by me, but for which I received supportive feedback during the design process. Due to the advisory contribution of the additional authors and supervisors, I use the term 'we' instead of 'I' in this dissertation.

The contents of the first paper are described in detail in chapter 4, the second paper is described in chapter chapter 5 and the third paper is described in chapter 6.

1.4 Thesis Structure

The remaining thesis is structured as follows:

First, chapter 2 presents the background and related work in the areas that are relevant to this dissertation. This includes an overview of work in SLAM tracking for HMDs, detecting hands with RGB and RGB-D cameras as well as multi-user virtual reality. Also, research work done on multi-user interaction and hand-body association is given.

Then, chapter 3 presents 'EasyHand', the underlying system that is used in all presented experiments to combine different hand tracking systems and render the results, as well as to create colocated scenarios and assign tracked hands to users.

Next, chapter 4 describes the first experiment to create colocated scenarios for SLAM-tracked headsets with hand tracking and compares it with other colocation methods.

The following chapter 5 presents a method for hand size estimation of the user and the usage of this information to do a 3D pose estimation for tracked hands with an RGB camera. Tracking accuracy and tracking range are compared with state-of-the-art tracking systems for VR.

For the final experiment, chapter 6 describes methods to assign tracked hands to virtual hands in a colocated multi-user VR setup as well as the experiment to determine the method with the highest accuracy.

Finally, the dissertation is concluded in chapter 7 and an outlook for future development and improvements is given.

CHAPTER 2

Related Work

This chapter provides an overview of the relevant previous research conducted in the central areas of this thesis. Research topics include SLAM tracking, multi-user VR systems, hand detection in VR environments, and hand-body association.

Before we go deeper into previous work on the specific research topics relating to this paper, we want to position our work in the reality-virtuality continuum of Milgram et al. [75]. They present a scale that can be used to classify whether an application takes place in complete virtual reality (VR), the virtual augments the real (augmented reality - AR), the real augments the virtual (augmented virtuality) or whether it takes place completely in reality. They define everything that mixes the real and the virtual as mixed reality (MR). Since our work deals with the area of colocated virtual reality and the recognition of real hands and their virtual representation (AR), we would see the overall context of this work in the mixed reality area, even if the main use case of this work takes place in a completely virtual space (VR).

Since we create colocated VR scenarios for SLAM-tracked VR headsets, we begin by introducing existing SLAM-tracked headsets, elucidating their operational principles, and addressing the research related to their limitations. Next, we examine hand detection in both VR and non-VR contexts. We explore research involving RGB depth (RGB-D) cameras and non-depth (RGB) cameras, highlighting their respective limitations and advantages. Furthermore, we review studies on 3D pose estimation employing these camera types. Given that our work involves evaluating the accuracy of our system, we also present existing research on the accuracy assessment of hand detection systems in static and dynamic interaction scenarios.

Then, we delve into the domain of multi-user VR systems, categorizing existing approaches and show work done in interactions within such systems. Additionally, we discuss research on colocation, with focus on SLAM headsets, as well as experiments done using external camera systems.

Lastly, we analyze the existing research on hand-body association, their limitations, and present the reasons why the existing work in this area is of limited use for our experimental purposes.

2.1 SLAM Tracking for Virtual Reality Headsets

SLAM, an acronym for 'Simultaneous Localization and Mapping', refers to a set of algorithms designed to detect the surroundings using one or more cameras or sensors, build a map of the tracked environment, and determine the position of an object, typically a robot, vehicle, or VR headset, within this environment [42]. This technology enables the tracking of objects within a confined space without the need for an external camera system, such as the HTC VIVE with its external Lighthouse tracking system, which is used to track objects in a physical space [81], or the Oculus Rift headset, which uses low-cost external micro-electromechanical systems sensors and a predictive strategy to minimize lag [57].

An example of SLAM tracking is illustrated in Figure 2.1, where a camera tracks environmental features, creating a point cloud, while the user is simultaneously positioned within the mapped environment.[1]

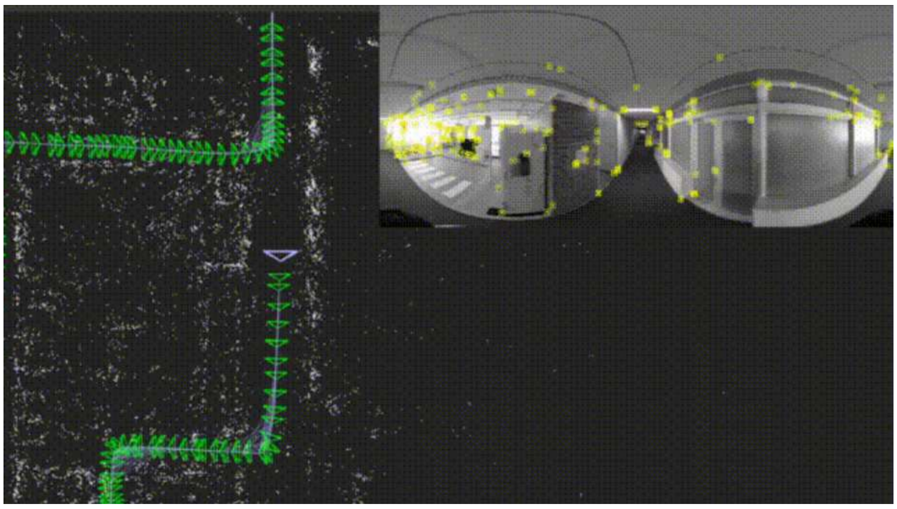


Figure 2.1: An example of SLAM tracking where features of the environment are tracked in a point-cloud an the user is positioned in the mapped environment.

SLAM technology finds extensive applications in mobile robots, improving the navigation of autonomous devices like robot vacuum cleaners [78][52]. It extends to a wide range of domains, including robotics [53][46], autonomous driving [11], and virtual reality headsets [9][72]. In our context, we focus on Visual SLAM, which employs images from cameras,

[1] `https://medium.com/@mccartneykyle12/whats-the-deal-with-s-l-a-m-and-tracking-technology-d1ef6ede9bbb` (Accessed: 2023-10-09)

such as RGB and RGB-D cameras, to achieve SLAM, as opposed to LiDAR SLAM, which relies on LiDAR sensors. Barros et al. conducted a survey of various Visual SLAM algorithms, providing an overview for developers to explore existing solutions and challenges in Visual SLAM research [65]. Similarly, Saputra et al. conducted a survey specifically on SLAM in dynamic environments, addressing three key problems: robustness, dynamic object segmentation and tracking, and joint motion segmentation and reconstruction [97].

An algorithm specifically designed for indoor environments and mobile RGB-D cameras was presented by Brunetto et al. [8]. They proposed a real-time 3D reconstructed environment that, while usable, operates too slowly (10Hz) for real-time applications in VR. In the context of virtual reality headsets, it is essential to have a fast and mobile SLAM technique. Additionally, since common commercial VR headsets are typically equipped with RGB cameras without the ability to track depth input, alternative algorithms are required for Visual SLAM with RGB cameras.

Williams et al. presented a mapping generation system for SLAM tracking with VR headsets, offering keypoint recognition for monocular SLAM within a 33ms frame time, improving the approach of Brunetto et al. and facilitating real-time VR scenarios [124]. Bustos et al. dived into the bundle adjustment technique commonly used in SLAM systems and its applications in VR tracking for 6DOF tracking and environment mapping, proposing further improvements to accelerate calculations and enhance the overall complexity of the SLAM system [9].

In our work, we use the Meta Quest and Meta Quest 2 headsets, that are employing Visual SLAM tracking with four cameras on the headset for inside-out tracking [72]. Building upon existing SLAM research, they leverage their 'Oculus Insight' AI technology to enhance SLAM, controller recognition, and 6DOF tracking for their headset [39]. This involves using motion capture data and device simulation in predefined scenarios as training data to improve tracking accuracy, along with presenting a computer vision architecture to optimize map generation and updating based on changes in the user environment. The result is a submillimeter tracking accuracy for the Meta Quest 2, even surpassing the accuracy of the HTC Vive Trackers 2.0, that use outside-in tracking technology [40].

2.2 Advancing Hand Detection

In addition to the conventional controllers that became standard with commercial VR headsets, hand tracking presents a natural alternative to translate users' hand movements into the virtual space, enabling intuitive interaction. For example, a comparative user study by Voigt-Antons et al. demonstrated that tracked hand interactions in a virtual environment enhance the sense of presence and offer improved usability for activities such as grasping and virtual typing [116].

Manresa-Yee et al. introduced a real-time algorithm to track and recognize hand gestures and emphasize the usability for interaction with video games. Their approach involved identifying various hand characteristics, extracting these features, and using a finite-state classifier to determine hand configurations [67]. Feuchtner explored the phenomenon of body ownership illusion in virtual and augmented reality interactions and its potential to transcend the limitations of our physical bodies, leading to more effective and engaging interactions [21]. Khundam et al. argued that hand tracking holds great promise in medical applications due to its naturalness in real-world scenarios and usability in contrast to conventional controllers, highlighting the ongoing importance of research in this area [48].

Hand recognition techniques can be broadly categorized into two areas: recognition using gloves worn by the user [133][110][82], and visual recognition employing cameras and sensors [43][41][37][130]. In 1986, the company 'VPL Research' developed the first commercial glove for hand tracking, which used glass fibers to detect finger curvature [133], as illustrated in Figure 2.2. Subsequent commercial data gloves, including the CyberGlove [47], emerged in the wake of this development. Temoche et al. introduced a low-cost glove designed for VR and commercial use [110], and other researchers sought to enhance glove tracking [50]. Brendan O'Flynn et al. presented a case of using data gloves for arthritis rehabilitation, introducing a smart glove equipped with sensors, processors, and wireless technology to quantitatively assess range of motion with the goal of improving the rehabilitation process, where recalibration is no longer required for each user [82]. Xu et al. proposed a collision detection algorithm that relies on predicting the movement direction of a virtual hand and that is effective in conjunction with a data glove, particularly in a mine training simulation system. They also suggested a dynamic algorithm to resolve collision detection issues between two solid simulation models [126].

Modern gloves frequently incorporate haptic feedback to enhance immersion [111][89]. Notable examples include the Senseglove Nova [2]), Haptx glove [3]), and TESLAGLOVE [4]). However, these devices, while enhancing presence, can be costly for end-users and require the user to wear a fabric device consistently.

Alternatively, there are tracking systems that optically track the user's hands using cameras and sensors. One such system is the LeapMotion controller, which relies on infrared cameras and emitters to capture hand movements [3]. The subsequent sections will delve into systems and research that employ RGB-D or simple RGB cameras for hand tracking.

2.2.1 Hand Tracking and Pose Estimation Using RGB-D Cameras

A large selection of different types of cameras is available, which can also be used for hand detection. One approach to hand tracking involves the use of RGB-D cameras. These cameras typically employ additional sensors, such as infrared sensors, to determine

[2]Senseglove Nova: `https://www.senseglove.com/product/nova/` (Accessed: 2023-10-17)
[3]Haptx: `https://haptx.com/` (Accessed: 2023-10-17)
[4]TESLAGLOVE: `https://teslasuit.io/products/teslaglove/` (Accessed: 2023-10-17)

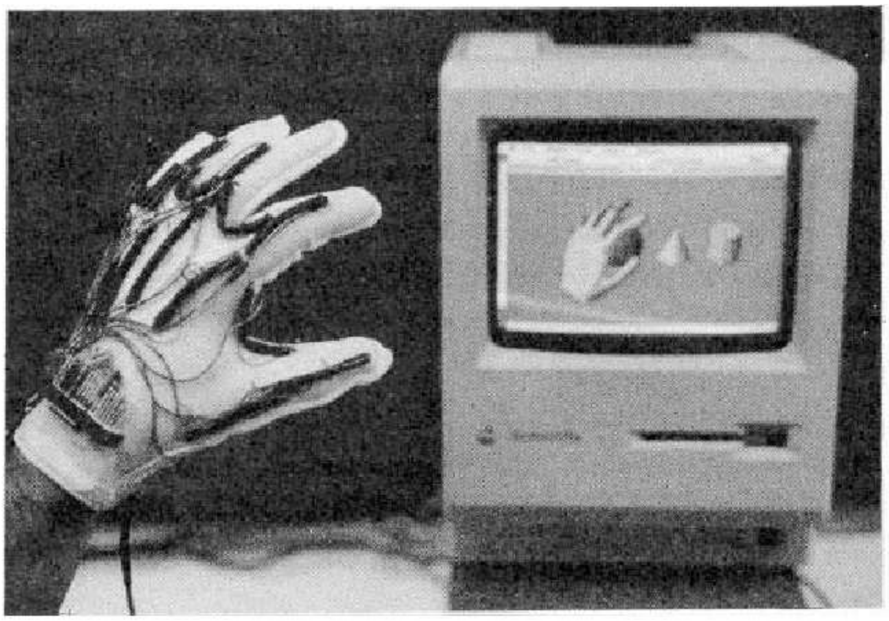


Figure 2.2: One of the first data gloves, developed by 'VPL Research' for interaction in virtual environments [133].

the depth of visually tracked objects in the image. Many different techniques exist[5]. Structured Light Cameras that use light projection and infrared sensors to perform pattern recognition for depth calculation can be very precise, but also require a long computing time. Time-of-flight cameras, on the other hand, calculate the time of flight of a light pulse from their infrared emitter for depth calculation. This means that depth data can be displayed in real time, but there may be a loss of precision. Additionally, there are the Stereo Vision Cameras, which calculate depth information by triangulating the images from two camera lenses. These are easy to replicate in terms of hardware design and are inexpensive, but the range can be limited and require long computing times.

Prominent examples of such cameras include the Intel RealSense Depth Camera D435 (Stereo Vision Camera), which is equipped with two stereo RGB cameras and an infrared sensor for depth detection of up to 10 meters [6]), or the ASUS Xtion PRO LIVE camera (Structured Light Camera), which features an RGB camera, two infrared detectors, an accelerometer, and a microphone [7]). Both cameras are depicted in Figure 2.3. RGB-D cameras find utility in various applications, such as 3D reconstruction using depth and

[5]Depth Camera Principles: `https://wiki.dfrobot.com/brief_analysis_of_camera_principles` (Accessed: 2023-07-01)

[6]Intel RealSense: `https://www.intelrealsense.com/depth-camera-d435/` (Accessed: 2023-10-24)

[7]ASUS Xtion PRO LIVE: `http://xtionprolive.com/asus-xtion-pro-live` (Accessed: 2023-10-24)

color images [114] or in real-time hand tracking, an area of particular interest for our use case.

Figure 2.3: Two RGB-D cameras are depicted. *Left:* Intel RealSense Depth Camera D435; *Right:* ASUS Xtion PRO LIVE

In a prior work by Frati et al., a commercial camera, the Microsoft Kinect, was utilized in combination with a wearable haptic device for hand tracking and rendering, allowing for hand interactions within a virtual reality environment [24]. While our system does not depend on additional hardware worn on the hand, this work highlights the necessity and potential of depth cameras in providing interactive hands within virtual scenes.

Huang et al. conducted a survey and performance analysis of hand shape and pose estimation techniques using RGB-D cameras [41]. They identified several state-of-the-art methods achieving low estimation errors (<10mm), enabling practical interactive applications. Despite their effectiveness and efficiency, certain challenges, such as occlusion and self-similarity, require further investigation. The authors emphasize the need for additional research to address issues in 3D pose estimation.

Researchers Sharp et al. [103] and Malik et al. [66] presented implementations for hand tracking and pose estimation with depth cameras. Both approaches use a single depth camera to estimate the pose of the tracked hand. Sharp et al.'s approach combined a multilayered discriminative reinitialization strategy for per-frame pose estimation with a model fitting process based on stochastic optimization of an objective function. Their evaluation yielded a tracking range of 0.5 to 4 meters and a precision of less than 3 centimeters [103]. Malik et al. improved upon this by generating a 3D mesh representation of a hand from a single depth image. They used a synthetic data set with accurate joint annotations, achieving a joint location error of less than 1.5 centimeters, segmentation masks, and mesh files derived from depth maps for neural network recognition. The processing time from the depth image to the mesh was 3.7 ms [66].

However, it is important to note that depth cameras are typically more expensive than RGB cameras (e.g. simple webcams). Consequently, these approaches are not always intended for widespread commercial use. On the contrary, most computer users possess webcams that lack depth estimation sensors, necessitating alternative methods to estimate the pose of tracked hands. The lower costs as well as the better availability is the reason why this work focuses on the use of RGB cameras. The following section will delve into works related to hand tracking and depth estimation with RGB cameras. Our proposed solution to this challenge is also presented in section 5.2.3.

2.2.2 Hand Tracking and Pose Estimation Using RGB Cameras

Images retrieved from RGB cameras consist of only three color channels - red, green, and blue. With this information alone, tracking hands and estimating their virtual 3D positions becomes more challenging in the absence of depth information. Consequently, much research has turned to machine learning-based classification techniques, utilizing information from the image detection process to derive 3D hand and joint poses [10][107][119][62]. Our solution to this challenge, explained in section 5.2.3, also incorporates additional information, specifically the real size of the user's hand, to transform tracked hands into the 3D world.

Before transitioning the tracked hand into three-dimensional space, an analysis of the 2D image to detect the hand in it must occur. For instance, Zariffa et al. achieved hand segmentation in egocentric video using pixel-wise skin classifiers, followed by shape-based post-processing techniques. This allowed hand tracking in a 2D space using a single camera [129]. Hammer et al. presented their own algorithm, also relying on contour detection. Due to their requirement for use in interactive environments, their detection process operates in real time [36]. Both systems identify the contour of the hand without the need for markers worn on the hand, but do not include finger joint detection.

The MediaPipe framework, as introduced by Lugaresi et al., offers perception pipelines for various tasks, including object detection, utilizing machine learning with RGB cameras [64]. Zhang et al. showed a hand tracking implementation with MediaPipe extending the base hand detection with a 2.5D joint recognition for more than two simultaneously visible hands with good performance and precision [130]. Due to its open accessibility and strong performance, especially on mobile devices with processing times ranging from 1.1 to 7.5 ms on iPhone 11, depending on the model used, we selected the MediaPipe framework as the foundation for our RGB-based hand tracking method. However, like many similar methods, Zhang et al.'s approach provides 3D finger joint positions relative to the origin in the middle of the hand (see Figure 2.4 [130]). The distance between the hand and the tracking camera remains unknown. However, this information is essential to enable reliable interactions in three-dimensional space.

Estimating the 3D pose of the hand is a widely researched area. For instance, Sun et al. proposed the first real-time one-stage method for pose estimation from a single RGB image, predicting 2.5D hand joint coordinates while locating two hand regions. Their experiments on public datasets demonstrated competitive results with state-of-the-art methods [107]. Zimmermann et al. published a method for 3D hand pose estimation based on RGB input, tested on a synthetic dataset for neural network hand recognition, with performance comparable to existing depth-based approaches [134]. Additionally, Lin et al. utilized a neural network-based pipeline for hand tracking and created a new 3D dataset to train the algorithm for two simultaneously tracked hands, reporting a mean

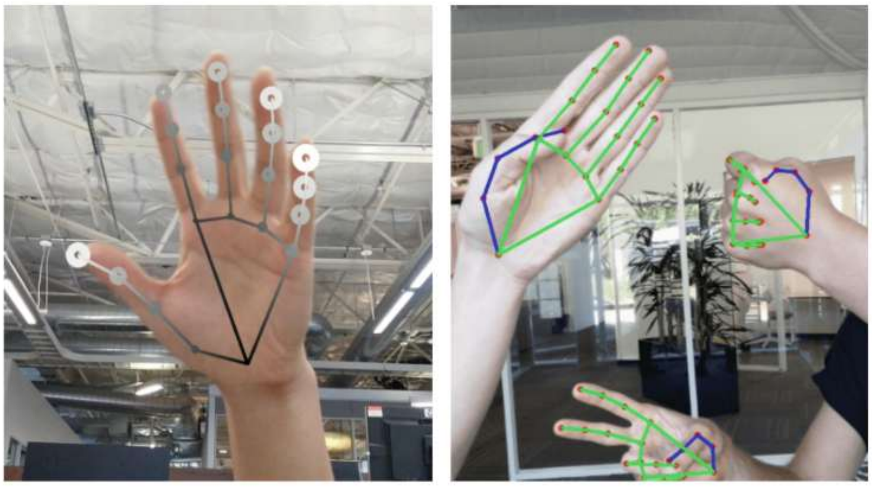


Figure 2.4: Tracked hands using the MediaPipe framework, which is used in our implementation. *Left:* Tracked finger joints with relative depth to the wrist. *Right:* Simultaneously tracked multiple hands.

End Point Error of 12.47mm, but only within arm's length range and for a maximum of two simultaneously visible hands [62].

Han et al. used four monochromatic cameras to track a maximum of two hands in 3D space for real-time virtual reality applications [37]. Their solution was subsequently implemented in Meta Quest headsets for hand tracking. Panteleris et al. achieved 3D pose estimation for hands tracked in 2D by using OpenPose for 2D hand recognition and non-linear least-squares minimization to fit a 3D hand model to the estimated 2D joint positions, thus recovering the 3D hand pose [87]. Che et al. presented a detection-guided method capable of recovering 3D hand posture with a color camera, by lifting the 3D pose from the estimated 2D joints through a model-fitting approach [10]. Wang et al. developed a method to track 3D real-time interactions using a monocular RGB camera. They employed a multi-task convolutional neural network (CNN) that regresses multiple complementary pieces of information, including segmentation, dense matchings to a 3D hand model, and 2D keypoint positions [119]. This approach also proposed intra-hand relative depth and inter-hand distance maps.

While several methods exist to track hands with an RGB camera and lift detected hands into 3D space, none of them fully suited our use case. They either are limited to tracking two simultaneous hands, operate solely in the 2D or 2.5D space, or do not track beyond the user's arm's reach. However, for our planned use within colocated scenarios it is necessary for the hands of several users to be recognized at greater distances and correctly positioned in three-dimensional space to enable interactions within the virtual environment. This led us to propose our solution for estimating the 3D pose of tracked hands. As explained earlier, we selected the MediaPipe hand tracking implementation by Zhang et al. as our base tracking system due to its open accessibility and good performance for our 3D pose estimation, as discussed in section 5.2.3.

2.2.3 Evaluating Hand Tracking Solutions

Several reports and evaluations have examined the precision and usability of common hand tracking systems, which is highly relevant to our work as we plan to compare these systems, namely the Meta Quest and Leap Motion Controller, with our implemented depth solution in section 5.3. With this we want to show that the proposed solution is suitable for use in colocated VR scenarios and has advantages over existing commercial solutions.

Let's first look at metrics that can be evaluated in hand recognition systems. The first metric to be mentioned is **accuracy**, which indicates how closely the tracked hand positions and rotations match the real positions and rotations. To determine this, a ground-truth comparison can be used, for example by an external tracking system, as used by Abdlkarim et al. or by us in this work [1]. The **latency** can also be used as a metric that measures the delay between a user's physical hand movement and the corresponding response. in the latency evaluation of modern HMDs, for example, Gruen et al. use external cameras that observe virtual and real inputs and measure the differences with a clock with submillisecond accuracy [33]. **Tracking range** is another important metric that we will discuss in chapter 5. This can also be measured, as in our work, by an external ground-truth tracking system, for example. Other metrics are **robustness**, to evaluate how robust the system behaves in difficult situations (e.g. covered hands or changing lighting conditions), **usability**, such as ease of calibration or user comfort, which is usually evaluated with user tests[20], or **performance**, in which the computational effort is measured [130].

For example, Schneider et al. reported the accuracy of finger tracking for touch-based tasks like pointing or drawing in the virtual environment, finding that Meta Quest and Leap Motion outperformed HTC Vive hand tracking in terms of spatial accuracy. Users generally preferred the Leap Motion sensor [98]. Their previous study also demonstrated superior tracking accuracy for Leap Motion compared to HTC Vive, with a Z error of approximately 2.6 cm for interactions with horizontally aligned surfaces (and 1 cm for vertically aligned surfaces) in walk-up-and-use scenarios [99]. Vysocky et al. conducted an accuracy study on Leap Motion, reporting an error of up to 1 cm for measurements taken at a distance of 20 cm [117], while Bachmann et al. performed an initial evaluation of the general usage and interactability of Leap Motion as a contact-free pointing device [3]. Even earlier, Weichert et al. reported that the Leap Motion controller could deliver near- and submillimeter precision when detecting hands, allowing developers to accurately visualize the pose of a user's hand inside a virtual environment [121]. Additionally, Mizera et al. compared the visual tracking accuracy of the Leap Motion sensor with the accuracy of two data gloves [76]. They found that the Leap Motion was less precise when measuring finger bending but very precise in estimating fingertip positions, making it the best device for fine manipulation tasks involving the thumb and opposing fingers.

A recent study of Matulic et al. propose a mirror-based solution that reflects the front camera of a smartphone to estimate the 3D position of the fingertip with the help of a deep neural network. They report a mean precision of 6mm and how their system could be adapted to VR systems in the future [69].

In the context of strategies to measure tracking accuracy in VR hand tracking systems, Abdlkarim et al. introduced a framework and demonstrated its application in the Meta Quest 2 [1]. Their setup involved a height-adjustable table that could be lowered to a height difference of 82 cm. The Oculus was attached to the top to track the user's hand during the experiment. Ground truth data was collected from infrared markers placed on the table and tracked by an optical camera system. The users were instructed to point at different markers during the experiment to collect data. The authors reported an average error of 1.1 cm in the position of fingertips for the Meta Quest 2. Ferstl et al. introduced and evaluated strategies to mitigate the impact of hand tracking loss, highlighting the potential influence of tracking loss on the user experience [20]. As a result, our evaluation design includes tracking loss distances for all hand tracking systems evaluated in section 5.3.2.

However, for VR hand tracking systems, standardized evaluation techniques for dynamic hand movements are lacking. Such techniques are commonly used in fields like machine tools, iGPS, and laser trackers. For example, Wang et al. tracked the trajectories of an industrial robot using a standardized strategy [120], and Ding et al. measured dynamic tracking error for five-axis CNC machining [17]. Due to the absence of standardized evaluation techniques for dynamic hand movements in VR, we have developed our own method, tailored to our experiment, to determine tracking accuracy and error curves for dynamic hand movements in section 5.3.2.

2.3 Advancing Multi-user VR Systems

By having social aspects such as communication, cooperation and collaboration, multi-user applications introduce a realm of possibilities that single-user applications cannot offer. Even in non-VR contexts, it has been demonstrated that multi-user virtual environments can facilitate enhanced learning experiences compared to traditional direct instruction methods [112].

When it comes to implementing multi-user experiences in the domain of virtual reality, developers have various options to consider. Podkosova has introduced a taxonomy that categorizes different types of such systems, which is depicted in Figure 2.5.

In contrast to single-user VR experiences, where each user typically occupies their own physical and virtual space, multi-user VR can be characterized by whether users share the physical space, the virtual space, or both. One common scenario is where users share the virtual space while maintaining their distinct physical spaces. This configuration is prevalent in multi-player VR games [8], virtual meeting environments that emphasize

[8]Tetra Studios: `https://www.tetrastudios.com.au/` (Accessed: 2023-10-10)

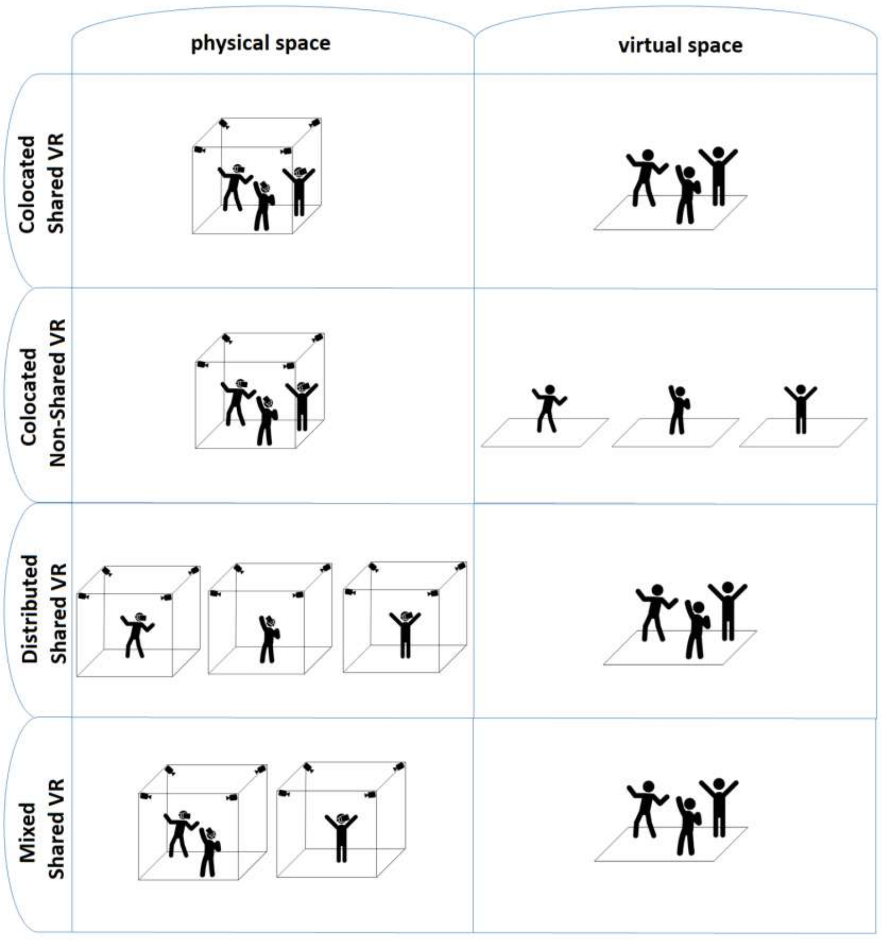


Figure 2.5: Taxonomy of multi-user VR created by Podkosova [92].

social interactions [9], and casual VR chat platforms [10]. The highly anticipated Metaverse is also expected to incorporate such multi-user VR access [115]. For example, Shi et al. used such an environment to demonstrate that a shared virtual environment can enhance communication efficiency in facility management [104]. You also don't run the risk of colliding with other players or having a mismatch between the physical and virtual presence of another user (e.g. hearing him behind you while standing in front of you). When using physical devices, however, a mismatch between physical space and physical device must be aligned, as proposed by Fink et al. with their Re-locations method [22].

Conversely, the colocated non-shared VR approach involves users occupying the same physical space while experiencing their unique virtual spaces. For example, the commercial VR experience 'YULLBE GO' offers free-roaming colocated non-shared VR experiences, where up to ten users share a physical space of approximately 80 square meters, each

[9]Spatial: `https://www.spatial.chat/` (Accessed: 2023-10-10)
[10]VRChat: `https://hello.vrchat.com/` (Accessed: 2023-10-10)

engaging with their own VR application asynchronously [11]. These experiences typically include collision avoidance mechanisms, with visual cues within the application signaling proximity to other users. However, effectively managing collisions while maintaining a high level of presence and immersion remains an ongoing challenge.

Various approaches have been explored in the creation of multi-user VR applications. As early as 1996, Benford et al. investigated shared spaces and provided an example of an internet foyer to showcase the possibilities offered by multi-user VR [5]. Just two years later, Frécon et al. introduced 'DIVE', a network architecture for creating distributed virtual environments. Although it was not developed explicitly for VR, it presents techniques that are applicable in this context [28].

Research in the area of multi-user VR has continued to evolve, often intersecting with other research domains. For example, Langbehn et al. playfully explored redirected walking in a colocated virtual reality scenario, where multiple users share the physical and virtual space. They introduce a technique that enables users to explore virtual spaces five times larger than the shared physical space [55]. Redirected walking in separate physical spaces and shared virtual spaces have also been studied by Xu et al., who developed methods to reduce the frequency of resets by predicting potential reset points based on a user's current reachable area [125].

The last approach in the spectrum of multi-user virtual reality is colocated shared VR, where all users occupy both the physical and virtual space collectively. This configuration introduces unique challenges in terms of implementation, as it involves the simultaneous management of physical and virtual experiences. These challenges will be examined in greater detail in the subsequent section.

2.3.1 Exploring Colocated VR

As described in the previous section, colocated shared VR refers to multi-user applications where multiple users share both the virtual space and the physical space. An example on colocated users can be found in Figure 2.6. Recent studies have shown that this type of multi-user experience can create a strong social atmosphere with a positive impact on social closeness between users [108].

Such a colocated scenario can be either symmetric, where all users use the same interface [15][91], or asymmetric, where users use different interfaces to interact with the shared environment [86]. For instance, Drey et al. conducted a comparative study of symmetric and asymmetric methods in the context of pair-learning and found that both approaches yielded equivalent results in terms of learning success [19]. In our work, we focus on symmetric approaches to give the users of the application a similar experience.

To reliably synchronize both realities, precise localization of all users in the physical space is essential. One common approach involves continuously synchronizing all users with the help of external camera systems. These camera systems are frequently used in

[11]YULLBE Go: `https://yullbe.com/yullbe-go/` (Accessed: 2023-10-10)

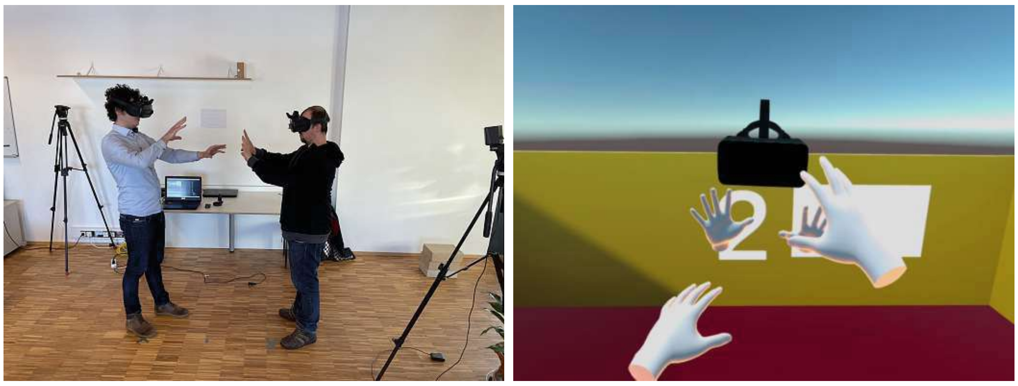


Figure 2.6: Two colocated users in both the physical and virtual space, with hand tracking enabled. The point of view (POV) is from the left user within the virtual scene.

VR arena experiences and are offered by companies such as Vicon [12] and OptiTrack [13]. However, these systems can be expensive and have limitations in terms of spatial size and the number of users they can accommodate.

As an alternative, Weissker et al. proposed a low-cost colocation system using the Lighthouse tracking system of the HTC Vive [122]. They use a single tracking system to synchronize the tracking data of multiple users to a common coordinate system, enabling colocation in a more cost-effective manner.

In contrast, SLAM-tracked VR headsets, such as the Meta Quest, use inside-out tracking without the need for external cameras. These headsets create their own internal environmental mapping of the physical space. Therefore, solutions are required to synchronize the physical positions of all users within their own environmental data.

Researchers have already developed solutions to address this challenge for virtual (VR) and augmented (AR) reality headsets. McGill et al. investigated practical and cost-effective ways to implement colocated scenarios for SLAM-tracked headsets. They categorized these solutions into two main types: *'Aligning to a single known point'* and *'Aligning to two known points'* [70].

The single-point calibration method is straightforward. It involves having a single point in the real world and a corresponding reference point in the virtual world. When the location of the user's headset relative to the real-world point is known, the user's virtual pose can be set accordingly. An example of this setup can be seen in the CAVE experience by Layng et al., where each user is assigned a seat with a predetermined position that is mirrored in the virtual world [58]. Herscher et al. employed a similar approach in their CAVRN system, where users are positioned in real-world seats corresponding to virtual orientations and positions [38]. However, these systems are limited as they only recognize

[12]Vicon: `https://www.vicon.com/hardware/cameras/` (Accessed: 2023-10-11)
[13]OptiTrack: `https://optitrack.com/applications/virtual-reality/` (Accessed: 2023-10-11)

users based on their seating positions, and further tracking and movement within the virtual world are not supported after relocalization. 'TritonVR' [14], a multiplayer shooting game, offers a different approach by allowing users to stand in the same physical location, followed by recalibrating their positions in the virtual environment to match their real-world poses. The precision of this method depends on how accurately users stand on the predefined positions.

On the other hand, the two-point calibration method uses two points or anchors in the real world to better approximate the users' poses and reduce drifting in larger rooms [70]. While this calibration method requires more calibration points and takes longer than single-point calibration, it can potentially provide more precise results by compensating for tracking errors. In our research, we implement both one-point calibration and two-point calibration in a standard room-sized environment (4x4 meters) and compare their results in section 4.4.

Colocation can also be achieved by tracking markers attached to headsets. DeFanti proposed a solution in which multiple users within a colocated space track each other using cameras on their HMDs to track ArUco markers attached to each user. These data are then shared among users to recalculate their relative positions in the virtual world [15]. However, this method depends on accurate and consistent recognition of each other's markers. We investigate a modified version of marker-based colocation in our experiments in section 4.2.2.

A drawback of the previously mentioned methods is their reliance on either additional AR-tracking cameras or the precise positioning of users in the real world. As more SLAM-tracked VR headsets, which also offer integrated hand tracking, become available (e.g., Meta Quest, or VIVE Focus[15]), we propose a solution in section 4.2.3 to use this feature for colocation. This approach ensures precision independent of the user and relies solely on the tracking system. Additionally, it eliminates the need for additional hardware, using only the SLAM-tracked headsets.

We can also differentiate between continuous calibration and one-time calibration. Continuous calibration, as the name suggests, updates each user's location every frame in the tracking system's coordinate system when external cameras are used. An example of this approach is the calibration using the Lighthouse tracking system of the HTC Vive by Weissker et al. [122]. Waller et al. used an infrared optical outside-in tracking solution with eight cameras and an HMD attached to a rendering computer worn by a user for their 'HIVE' system [118]. DeFanti used cameras attached to the users' HMDs to continuously track other users [15]. Furthermore, Podkosova et al. created ImmersiveDeck, a system that combines inside-out head tracking with motion capture, enabling multiple users to move freely in a large area (i.e. 200m^2) [91]. In the field of robotics, a shared spatial map can be used for collaborative applications to maximize efficiency in environment

[14]Triton VR: https://www.tetrastudios.com.au/tritonvr (Accessed: 2023-10-12)

[15]VIVE Focus: https://enterprise.vive.com/de/product/vive-focus/ (Accessed: 2023-10-12)

exploration [23]. Unlike continuous calibration, one-time calibration aligns all devices only once.

The accuracy of colocated systems after calibration depends on the tracking accuracy of the device. If the error becomes too significant, a recalibration is necessary. For instance, McGill et al. used one-time calibrations for their experiments, whether for a one-point or two-point calibration [70]. In seated experiences, such as CAVE or CAVRN, users are placed in real-world seats that correspond to virtual orientations and positions, and continuous calibration is not used [58][38]. Given our desire to minimize additional hardware and efforts, our approach relies on one-time calibration, leveraging the low drift of our SLAM devices after calibration. However, we ensure that recalibration remains a viable option with minimal effort.

2.3.2 Hand Interactions in Multi-User Systems

Interaction stands as a very important concept within the domain of virtual environments, such as in virtual reality, augmented reality or even for tabletop. It can be manifested through various means, including gestures and direct manipulation.

There is existing work comparing hand interactions to conventional controller input. Khundam et al. compared interaction time and usability between hand and controller interactions in intubation training, with the result that there are no significant differences [49]. Zhao et al. also found comparable user preference and performance between hand and controller interactions for virtual locomotion [131]. However, Schäfer et al. found significantly better performance and accuracy for interactions in which objects are picked up and put down between the use of hands and controllers [101]. This is also inline with other studies by Masurovsky et al. and Hameed et al. that show the preferential usability and accuracy of controller-based interactions compared to hand interactions [68][35]. However, they also emphasize that growing acceptance and familiarization with hand interaction systems, as well as the continuous improvement and better availability of hand recognition systems and their naturalness, offer the potential to overtake conventional controllers in terms of user experience in the future [68].

Extending the interaction concept into shared virtual worlds introduces both challenges and opportunities. For instance, as early as 2007, Streuber et al. investigated the impact of multi-user environments on joint actions and social behavior. In their study, they designed a test in which a colocated user pair had to navigate a stretcher through an obstacle course. Their findings suggested that humans can quickly adapt to the lack of haptic and tactile feedback when fully immersed in the virtual environment [106].

Further research has focused on designing frameworks and strategies tailored to multi-user contexts. For example, Gong et al. conducted a case study to develop interaction design strategies for multi-user virtual reality systems in manufacturing settings, underscoring the importance of robust interaction systems in such scenarios [31]. Langbehn et al. cautioned against making substantial user manipulations, like altering a user's voice pitch, as it may distract users from interactions in the virtual environment [54].

In addition to multi-user VR applications, Zaman et al. aimed to create a platform that enables collaborative tasks regardless of the type of head-mounted display, thus enabling collaboration between multiple VR and AR devices [128]. Olin et al. explored cross-device interaction in a virtual environment, allowing handheld device users to engage with VR headset users. Their research provided a framework and design suggestions for such scenarios, demonstrating immersive integration of non-VR users into virtual interactions [83].

Within the scope of multi-user AR applications, Jansen et al. introduced ShARE, a head-worn AR device equipped with a projector to enable simultaneous interactions with non-AR users. Through marker detection, outside users can interact with the projected image, fostering multi-user interactions across three different applications: collaborative games, competitive games, and external visualizations, resulting in a proof-of-concept device for multi-device multi-user AR interactions in a shared virtual world [44].

However, in the context of multi-user VR applications, one significant aspect of interaction involves the hands themselves, which is particularly relevant to the current work aimed at improving hand recognition and interaction in colocated scenarios. Pretto et al. conducted research on hand interaction through gestures using Google Cardboard as an HMD and Leap Motion for hand tracking. They applied this technology in a forensics use case, integrating real-world objects into a VR environment [93].

Beyond the VR domain, multi-user hand interactions have been explored in the media domain using multimedia tabletops. Del Bimbo et al., for instance, created a computer vision-based system and algorithms that uses hand recognition to enable multi-user interactions at a display table, even with distinguishing hands of two users that were overlapped in the 2D area [16]. Dohse et al. developed a tabletop that utilized a combination of computer vision for hand recognition and touch detection to distinguish hands in close proximity. Multiple users interacting with the tabletop can be seen in Figure 2.7. This work emphasized the importance of distinguishing and assigning hands to different users to enable interactions in multi-user and close proximity scenarios [18]. We address this issue in the context of colocated multi-user VR in chapter 6.

In the realm of colocated VR scenarios, an area of focus in this work, researchers have investigated various interaction possibilities and their impact on the user experience. Salzmann et al. examined collaborative assembly tasks using only hands in the automotive industry, demonstrating their suitability for two simultaneous users [96]. Their findings revealed higher precision in task execution and user preference for these tasks. Li et al. conducted user testing to evaluate the influence of interactions using hand-held controllers in multi-user co-location scenarios for cultural heritage. Their conclusion highlighted the positive effects of social influence on performance expectancy and effort expectancy [61].

All of these systems emphasize the importance of interactions in shared virtual experiences and the challenges they involve, such as obscured hands and hand-body ambiguity. Consequently, the system presented in this thesis represents a significant step towards providing reliable hand interactions in a colocated multi-user application.

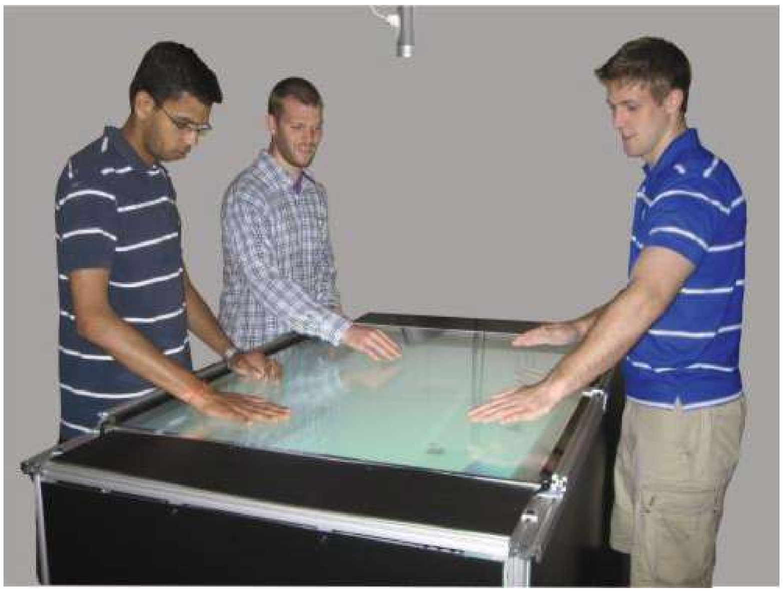


Figure 2.7: Three users interacting with the multi-touch tabletop by Dohse et al. On top is a camera attached that is used for hand tracking [18]. The image is taken the authors' work.

2.4 Fostering Hand-Body Association in Multi-user Scenarios

Assigning tracked hands to specific users in a colocated shared virtual environment is crucial for creating a seamless and immersive experience. In 2D space, various research studies have explored hand ownership in egocentric views to assign detected hands to users. For example, Frati et al. utilized the capabilities of the Kinect in conjunction with wearable haptic devices for hand tracking (as stated in section 2.2.1), making it easier to assign these hands to users due to the additional full-body tracking provided by the Kinect [24]. However, most tracking systems are limited to tracking only the hands of users, and the user's full body may not always be in the tracking device's field of view, which requires alternative solutions for hand assignment.

Narasimhaswamy et al. employed neural networks to associate hands with bodies by utilizing image detection of bodies and hands. They used overlapping tracking bounding boxes from image detection, as well as the locations of heads and hands, to perform hand assignments [80]. This approach demonstrates the potential for improved hand tracking and hand contact estimation with hand-body association. However, it relies on having a user's body within the camera's field of view for bounding box creation, as shown in Figure 2.8, an image taken from the work of Narasimhaswamy et al. [80]. Tsutsui et al. also used visual cues provided by image tracking, such as color, depth, skin texture, and shape, for hand identification in egocentric views [113]. However, their approach focused solely on identifying a user's own hands, without considering other users and their hands in the environment. Furthermore, their approach demonstrated a verification error rate of 36%, making it impractical for real-world scenarios and not applicable to our scenario.

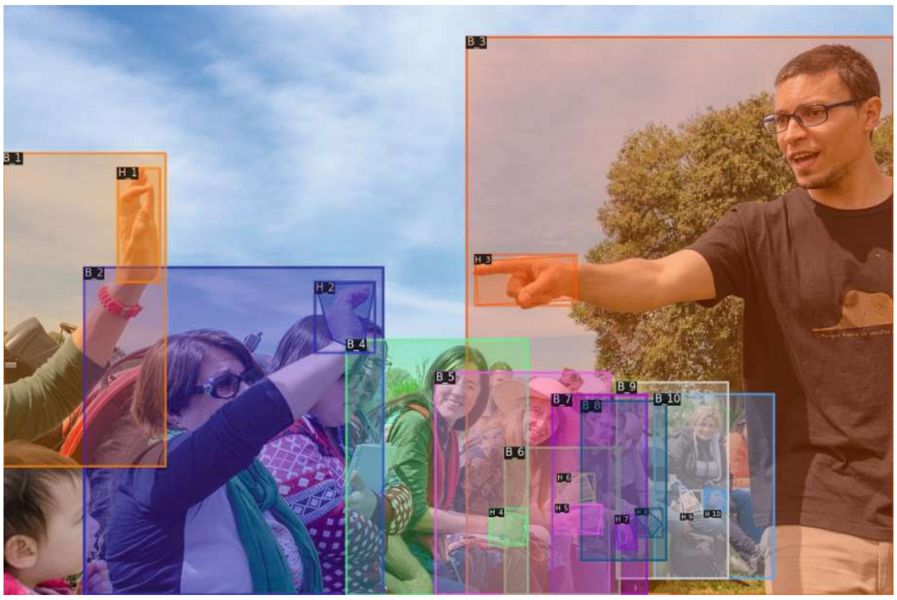


Figure 2.8: Hand-Body association by Narasimhaswamy et al. involves the creation of 2D bounding boxes to assign hands to users in the detected image [80].

Lee et al. explored the distinction of multiple hands in an egocentric view by utilizing the location of the hand in the two-dimensional recognition frame image. However, this method was not designed for three-dimensional tracked hands [59]. Lin et al. proposed a method for detecting hand raising in classrooms using image detection on a single image and a selection algorithm, achieving a mean Average Precision of 90% [63]. Building on this method, Zhou et al. extended it with an improved detection algorithm, a pose algorithm, and a matching technique for the raised hand and the hand raiser by utilizing the location of the hand and various keypoints of the hand-raisers body (face and limbs) with an average recognition precision of 83% [132].

However, all the mentioned research is either confined to hand assignment in the 2D space [59][63][132], or relies on additional tracking information such as skin color or 2D tracking boundaries [80][113]. These approaches are not directly applicable to our use case, which involves assigning hands in a colocated virtual environment in 3D space, with limited information available by the employed tracking systems, specifically, the location of the hands and user heads. Therefore, our methods presented in chapter 6 build on existing 2D domain methods and extend them with our ideas tailored to the 3D domain.

CHAPTER 3

EasyHand - A Modular Hand Interaction and Visualization Framework for Single and Colocated VR Scenarios

This chapter introduces ’EasyHand’, a Unity3D framework developed to harmonize the hand tracking capabilities of various hand tracking systems. ’EasyHand’ serves as the foundational system for visualizing, positioning, and interacting with tracked hands in all the experiments presented in this thesis. To maintain a concise and easily comprehensible structure within the chapter, the sections will typically commence with an outline of the concept and design of each component. If required, we will then proceed with an explanation of its implementation.

We begin by providing an overview of the ’EasyHand’ system, outlining its primary objectives and the requirements that guided its design and development. We will also elaborate on the hardware and software compatibility of this framework. We demonstrate how the unification of different base tracking systems within the framework for universal usage in the development process is done. This unification includes the integration of hand detection and gesture recognition events. After that, it is explained, how visualization for skeletal and mesh rendering as well as interaction is done based on the unified data.

Then, we address the conceptualization and implementation of the network layer, a mandatory component for creating colocated scenarios. This will be followed by an exploration of our efforts to enhance the framework’s capabilities through the implementation of plugins. Finally, we conclude this chapter with an overview of the current state of the ’EasyHand’ framework and discuss potential future development.

3.1 System Overview

The framework was initially designed with specific capabilities in mind. It not only serves as a hand-interaction layer, but also encompasses hand tracking capabilities. These capabilities encompass networking features for multi-user applications, including colocation capabilities, as well as the ability to assign tracked hands to virtual users. To our knowledge, the 'EasyHand' framework stands as the sole system currently available capable of creating colocated multi-user applications utilizing hand interactions. Moreover, it is compatible with various existing base hand tracking systems.

3.1.1 Overall System Design

Initially, the system was designed to unify various hand tracking systems, such as Meta Quest, LeapMotion, or HTC Vive, providing developers with an easily accessible framework to develop applications across multiple platforms. Each of these systems is equipped with its own unique API and functionalities for visualization, gesture recognition, and interaction (see Figure 3.1). The term 'gesture' used in this paper refers to a fixed hand posture, such as all fingers being bent ('Fist' gesture) or the index and middle fingers being stretched ('Peace' gesture). Our 'EasyHand' framework serves as an intermediary layer between the application and these base APIs, allowing developers to employ a unified system for visualization, interaction with the virtual environment and gesture recognition across all supported APIs, as illustrated in Figure 3.2. Furthermore, the system is designed to facilitate the seamless addition of new hand tracking systems to expand its compatibility.

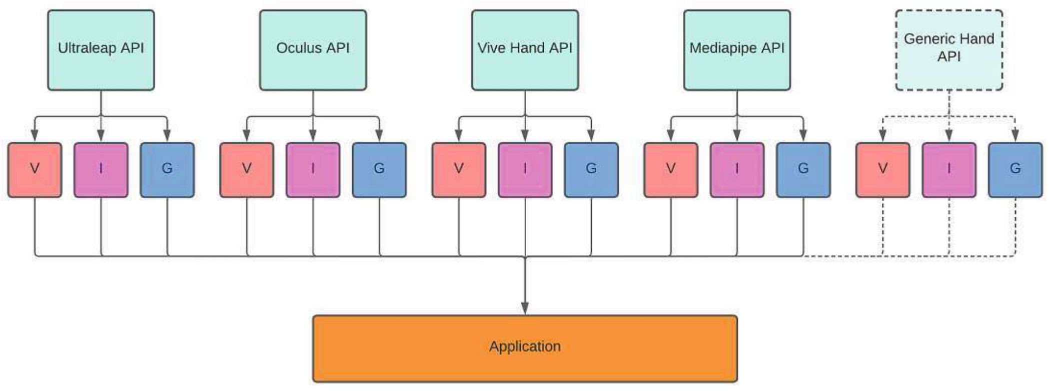


Figure 3.1: Overview of hand tracking API integration of different systems in one application. Each system comes with distinct visualization, interaction and gestures.

To fit our needs we defined the following requirements for the system:

- *The system should support several base systems.*
  As the framework serves as an intermediary layer for developers, it is designed to

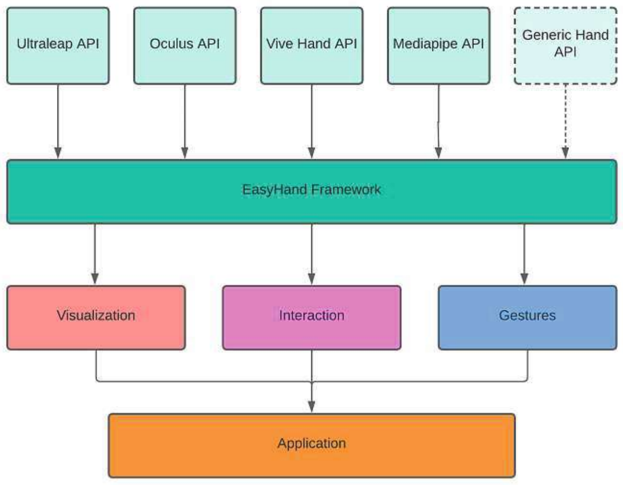


Figure 3.2: 'EasyHand' acting as a layer between several hand tracking APIs and the application, unifying visualization (V), interaction (I) and gesture recognition (G).

support several popular state-of-the-art hand tracking systems, including those used in the Meta Quest and Vive HMDs, as well as the Ultraleap system.

- *The system should implement visualization, interaction, and simple gestures. All data are unified.*
  Developers will utilize this system to craft interactive applications. Consequently, the framework should support a unified visualization for both, with simple lines and a visually more realistic mesh rendering. Furthermore, interactions should remain consistent across various base systems, with a similar set of recognized gestures (e.g. pinch) being supported. The system should also provide events for the detection and loss of the hand to adequately handle the visibility of the virtual hand. Additionally, all tracked data should be accessible through a unified data structure.

- *The system should be easily expandable.*
  As developers may have diverse requirements, they should have the flexibility to easily integrate hand tracking systems that were not initially included. This can be achieved by adding a unification class as an intermediate layer between the tracking system and the 'EasyHand' framework.

- *The system should be usable in real time.*
  Provided that the basic recognition systems allow it, a framerate of at least 60 frames per second (FPS), which is necessary in VR for a smooth run, should be achieved.

With the refinement of experiments and the addition of desired colocation capabilities, the following requirements were subsequently defined:

- *The system should support networking.*
  To create multi-user scenarios with hand tracking capabilities, the system must be capable of efficiently synchronizing all tracked hand data over a low-latency network layer.

- *The system should be able to create colocated scenarios.*
  The 'EasyHand' framework, utilizing integrated hand tracking solutions, should enable the creation of colocated multi-user scenarios, including the handling of calibration and synchronization of the coordinate systems.

- *The system should provide hand tracking for more than two hands.*
  To support other tracking systems inside colocated multi-user scenarios, the system should be able to track more than two hands, respectively the hands of colocated users. To achieve this the framework should incorporate a fitting RGB camera hand tracking algorithm and seamlessly integrate it into the system. This also enhances accessibility and hand tracking is not restricted to VR environments but can also find usage in desktop applications.

- *The system should be able to assign hands to virtual colocated users.*
  To be able to support each other's hand tracking by combining tracking cameras and ensure accurate interactions, the system should possess the capability to consistently assign tracked hands to virtual users within a colocated multi-user scenario.

3.1.2 Implementation Environment

Many different engines are available to develop applications for XR. Among them are the Unreal Engine[1], Unity3D[2], CryEngine[3] or OpenXR[4]. To support a wide range of hand tracking systems, it is necessary to choose a well-supported engine. A 2019 TIGA study in the UK indicates that a majority (72%) of developers surveyed use the Unity3D engine [105].

Due to the wide adoption and strong developer support for this platform, as well as the fact that we don't have to worry about rendering and physics, Unity3D was chosen as the development platform. For the basic hand detection of the different systems, the corresponding Unity3D plugins were included. This allows the 'EasyHand' framework to be made available to a wide range of developers.

[1]Unreal Engine: `https://www.unrealengine.com/en-US/xr` (Accessed: 2023-10-06)
[2]Unity3D Engine: `https://www.unity.com/solutions/vr(Accessed:2023-10-06)`
[3]CryEngine: `https://www.cryengine.com/` (Accessed: 2023-10-06)
[4]Khronos OpenXR: `https://www.khronos.org/openxr/` (Accessed: 2023-10-06)

3.2 Hand Detection

The most critical component of the system is hand detection itself. To provide a visual representation of the process, we start with integrating base tracking systems. This involves unifying the incoming base data into a standardized dataset with an index mapping of all detected finger joints that can be used independently of the tracking system in use. This unification of the base system is also where developers can add new tracking systems to the 'EasyHand' framework, as depicted in Figure 3.13. The developer only has to provide the index mapping (as explained in section 3.2.2) and an implementation of the 'ITracker' interface class, where the tracking systems low-level hand detection is mapped to the unified data set of 'EasyHand'.

From this unified dataset, we can proceed to use low-level events or, in cases where they do not already exist, create custom events to trigger actions when hands are detected or lost by the tracking device. Additionally, we employ a similar approach to handle detected hand gestures.

3.2.1 Low-Level Tracking Modules

As low-level tracking modules we chose systems that are widely used for optical hand tracking in XR scenarios. This includes the following systems (MediaPipe for RGB tracking is explained in chapter 5):

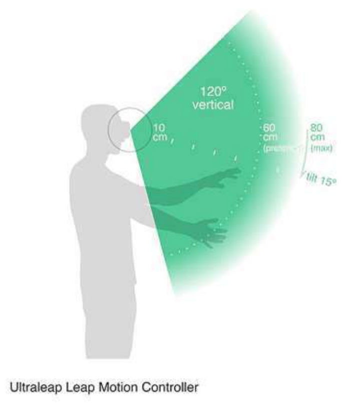
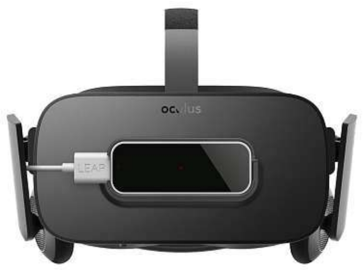


Figure 3.3: Hemispherical area of the interaction zone of the LeapMotion Sensor.

LeapMotion The LeapMotion hand tracking from Ultraleap is a tracking device consisting of two monochromatic IR cameras and three infrared LEDs. It can detect hands in a hemispherical area with an interaction zone of up to 60 cm and a field of view (FOV) of 120° x 150° [43] (see Figure 3.3)[5]. It is either used on a tabletop, on top of a

[5]LeapMotion Hemisphere: https://cms.ultraleap.com/app/uploads/2020/09/ultraleap-hand-tracking-interaction-zone-side.jpg (Accessed: 2024-01-07)

screen or it can be attached to a VR HMD (see Figure 3.4).
The underlying Ultraleap API extracts 27 distinct joints per hand with positional and rotational data (including elbow recognition which is irrelevant for our framework). The detected joints can be seen in Figure 3.4.



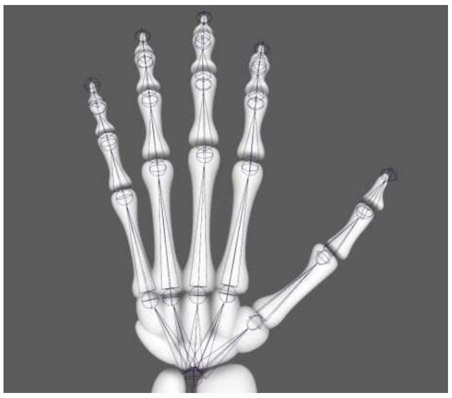


Figure 3.4: *Left*: The LeapMotion controller attached to a VR HMD. *Right*: Detected joints of the Ultraleap API

Meta Quest The Meta Quest[6] (2019) / Meta Quest 2 (2020) is a state-of-the-art virtual reality head-mounted display (HMD) that utilizes SLAM (Simultaneous Localization and Mapping) technology for 6 Degrees of Freedom (6 DoF) tracking. The device is equipped with four monochronous cameras, which are used to track the user's hands within a 3D space, a concept presented with the commercial solution MEgATrack proposed by Han et al.[37]. This hand tracking functionality is achieved through the application of deep neural networks, enabling the estimation of the hands' location and the detection of 21 individual hand joints. See Figure 3.5 for a visual representation of the device and how tracked hands appear in a greyscale passthrough view.[7]

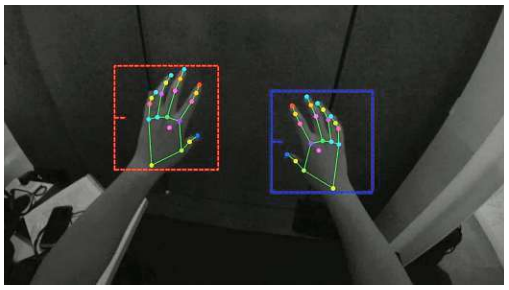
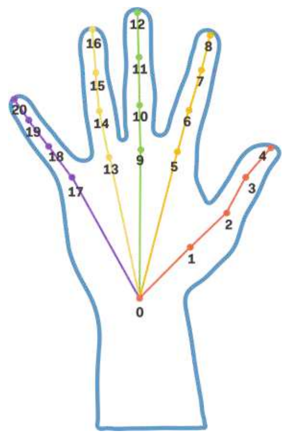


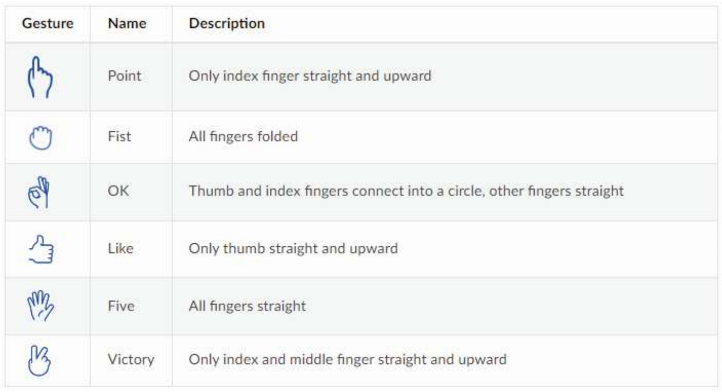
Figure 3.5: *Left*: The Meta Quest 2 HMD. *Right*: Detected hands and joints with the Meta Quest 2.

[6]Originally Oculus Quest before Meta rebranding in 2021

[7]Quest Hand Greyscale: `https://www.uploadvr.com/content/images/2021/04/QuestHandTrackingGreyscale.png` (Accessed: 2023-10-06)

Vive Hand Tracking The Vive Hand Tracking SDK provides hand tracking capabilities for supported headsets within the HTC Vive series, including the Valve Index. This SDK offers 2D and 3D tracking for a total of 21 finger joints and operates at a smooth frame rate of 60 FPS for minimal GPU requirements for VR with a NVIDIA GTX1060/AMD RX480 GP [14]. Additionally, the Vive Hand Tracking SDK includes support for base gesture recognition, which includes a set of six gestures, all of which are integrated into the 'EasyHand' framework. While the number of finger joints is the same as with the Meta Quest hand recognition system, the indexing differs. An overview of the detected joints and available gestures can be found in Figure 3.6.





Gesture	Name	Description
	Point	Only index finger straight and upward
	Fist	All fingers folded
	OK	Thumb and index fingers connect into a circle, other fingers straight
	Like	Only thumb straight and upward
	Five	All fingers straight
	Victory	Only index and middle finger straight and upward

Figure 3.6: *Left*: The detected joints of the Vive Hand Tracking. *Right*: Recognized Gestures of the Vive Hand Tracking.

3.2.2 Unified Joint Mapping

As depicted in Figures 3.4, 3.5, and 3.6, each of the base systems is capable of detecting finger joints and assigning a specific integer index to each joint. However, it's important to note that this indexing is not standardized across these systems. The ability to discern which part of the finger is currently being tracked is crucial for accurate hand visualization and gesture recognition.

To address this challenge, we introduce our own index mapping system (where we were inspired by LeapMotion and Quest) within the 'EasyHand' framework. This mapping defines a name for each part of the finger and assigns a corresponding index to each part. This name definition can be seen in Figure 3.7. During an initialization step, we establish a correlation between the index assigned by the base tracking system and the corresponding 'EasyHand' joint/bone index.

For instance, let's consider the 'EasyHand' joint named 'Wrist'. Initially, Vive and Quest assign this joint the index *'0'*, while LeapMotion assigns it an index *'20'*. Through the aforementioned mapping process, we ensure that we consistently obtain the correct location and rotation data when referencing the bone identified as 'Wrist.'

0. Wrist	10. Middle2
1. Thumb1	11. Middle3
2. Thumb2	12. Middle4
3. Thumb3	13. Ring1
4. Thumb4	14. Ring2
5. Index1	15. Ring3
6. Index2	16. Ring4
7. Index3	17. Pinky1
8. Index4	18. Pinky2
9. Middle1	19. Pinky3
	20. Pinky4

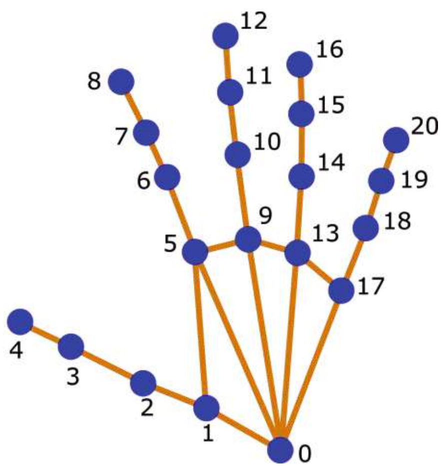


Figure 3.7: Joint index mapping and description of an 'EasyHand' hand skeleton.

With this mapping in place, we can easily maintain unified data structures for tracked hands. As one of the initial steps in the 'EasyHand' update cycle, we map the detected base joints to this unified data structure. This allows us to subsequently build upon these data, enabling us to perform all logic related to gestures, visualization, and interaction without relying heavily on the specifics of the underlying base tracking system. The current implementation is limited to the joints used in Figure 3.7. If a base system has more, the most suitable ones are selected and the others are discarded. The case that the base system does not have these joints is not yet covered, but offers the potential to calculate them in a future extension by interpolation, for example. However, this case has not occurred for the base systems used in this work. In this context, the only remaining information we need to obtain from the base system, if available, is about detection events and gesture recognition.

3.2.3 Gesture Recognition - Design and Implementation

Recognizing various hand gestures is a broad topic for scientific research in its own right. Therefore, our primary focus does not encompass the recognition of complex gestures but centers on the identification of simple gestures to facilitate basic gesture integration for developers. These simple gestures comprise six distinct hand gestures, as previously detailed in the HTC Vive hand tracking plugin overview presented in Figure 3.6.

To determine whether a specific gesture has been executed, we initially verify if the underlying recognition system successfully identifies any of the predefined gestures autonomously. In cases where this recognition does not occur, we resort to our custom implementation for gesture recognition. Since all of the mentioned gestures hinge on the extension or bending of fingers, we evaluate the angles at each finger joint. Each finger consists of four joints, forming three vectors. As illustrated in Figure 3.7 for the index finger, we can establish vectors as follows: $v_1 = \bar{65}$, $v_2 = \bar{76}$ & $v_3 = \bar{87}$. We then calculate the resulting angle (α) between v_1 & v_2, as well as between v_2 & v_3, add them up, and

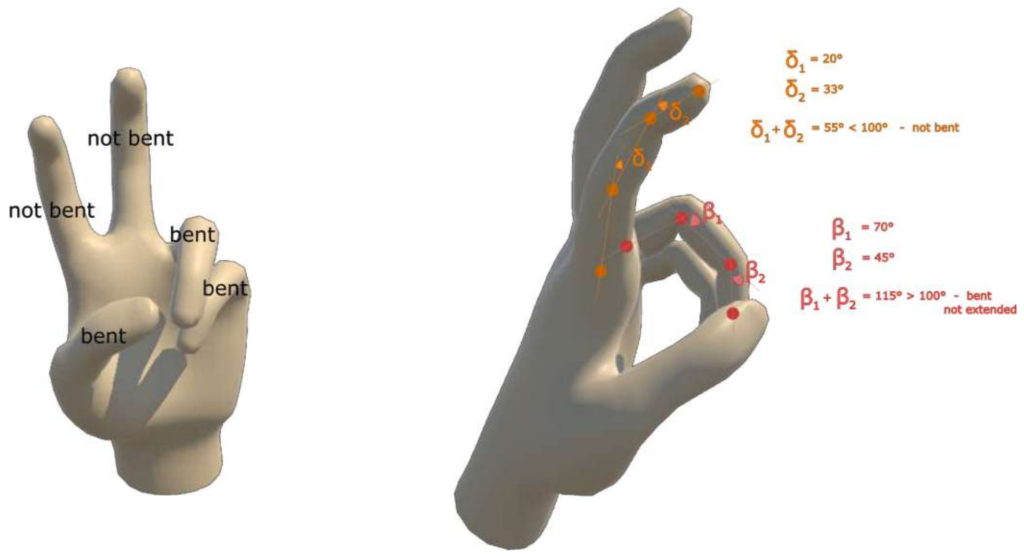


Figure 3.8: The 'Peace' gesture recognized by the system. *Left:* Labels indicate which fingers are bent and which are not. Here the system recognizes the gesture when all fingers are bent except the index and middle Finger. *Right:* Angle calculations between the joints are used to determine which fingers are bent and which are not.

subsequently apply predefined threshold values to discern whether the finger is extended ($\alpha < 45°$) or bent ($\alpha > 100°$). An example calculation for the 'Peace' gesture is shown in Figure 3.8, where all fingers need to be bent except the index and middle fingers.

3.3 Visualization and Interaction

After detecting the hands and assigning IDs to the detected joints, users require the ability to incorporate these hands into the virtual world. To accomplish this, it is essential to provide users with a means to visualize these hands and imbue them with physical capabilities, e.g. colliders, enabling interaction with the virtual environment. Special events that originate from the base tracking system when hands are newly detected or detection is lost are used to trigger visualization.

For visualization, we have implemented two widely recognized approaches (such as those used by Metzner et al. [74]): one employs an abstract representation of the detected joints and connecting bones, while the other utilizes a more realistic mesh representation of the hand. As Ricca et al. suggested that the visual representation of the hand does not always impact tool-based motor tasks training in immersive VR simulators, these visualizations promise good and interactively usable representations of the hand[95].

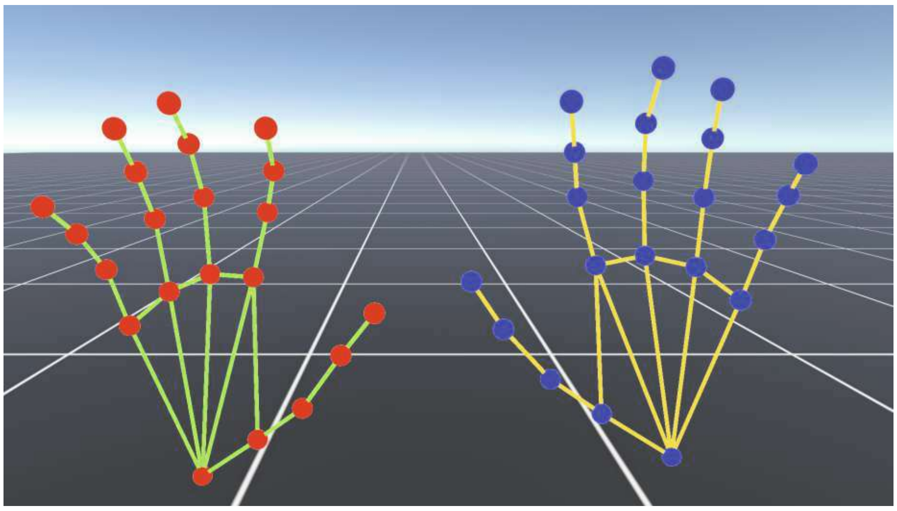


Figure 3.9: Skeletal rendering of detected hands.

3.3.1 Skeletal

One of the fundamental representations of the hand is the skeletal visualization, comprising spheres positioned at the detected joint locations and connecting lines to illustrate the bones. This approach enables users to identify each individual finger joint, as illustrated in Figure 3.9. The hand's world position is determined by the location of the 'Wrist' joints, and its orientation is derived from the resulting vector connecting the 'Wrist' and 'Middle1' joints.

3.3.2 Mesh

For a more lifelike representation of the tracked hand, we employ predefined meshes that offer adaptable visual presentations, allowing us to choose between realistic textures, abstract wireframes, and the ability to alter skin color, among other options.

To align the hand mesh with the detected hand, each rigged bone in the mesh is mapped to the corresponding joint index, as illustrated in Figure 3.7. The position and rotation of each joint are then dynamically adjusted through code to accurately mimic the connections of real hand bones. To ensure that the mesh closely matches the user's real hand size, we compute an ideal finger length based on the positions of the detected joints. This approach allows us to resize the bones to approximate the user's specific hand dimensions. The resulting visual representation can be observed in Figure 3.10.

We have taken advantage of existing resources, incorporating pre-assembled meshes from the 'OculusIntegration' Unity package [73], and we have also utilized the mesh calculations already used in the Vive hand tracking system for Unity [13].

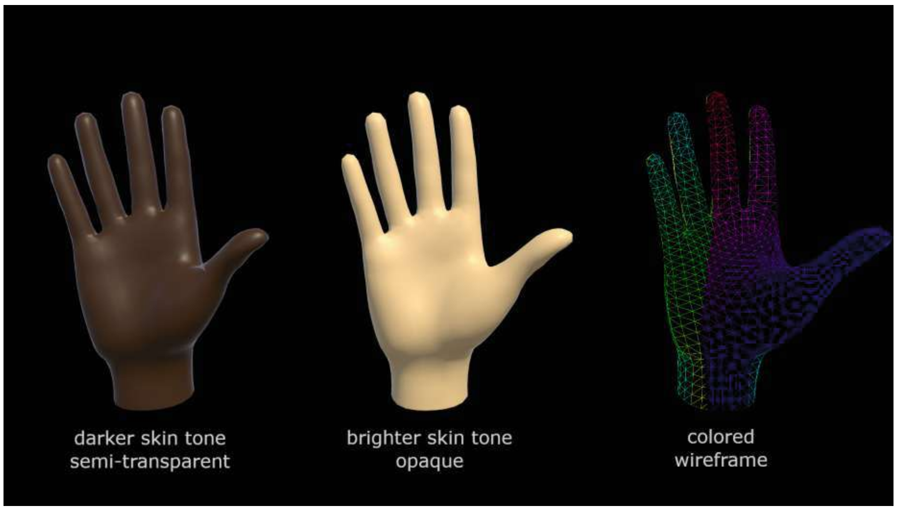


Figure 3.10: Skeletal rendering of detected hands.

3.3.3 Direct 3D Object Manipulation in Unity3D

Interacting with the virtual environment constitutes a crucial aspect of any VR application. One mode of interaction involves using predefined gestures, as explained in section 3.2.3. Another approach is direct interaction with and manipulation of the virtual environment, encompassing actions such as pressing buttons, picking up and tossing virtual objects, or selecting items. To enable these interactions, we must facilitate physical interactions with the visualized hand.

To achieve this, we used the existing collider component within the Unity game engine. Since we already have the locations of each finger joint, and therefore the length of each bone, we can easily generate capsule colliders with corresponding lengths for each bone. The positions of these colliders are then dynamically adjusted in each physics update loop to align with the user's hand movements.

An alternative approach would involve creating an exact mesh collider to match the hand precisely. While this approach would offer greater precision in interactions, it presents challenges due to the static nature of meshes in the Unity engine. Creating such a mesh every time the hand pose changes would impose an exceedingly high computational load. With our implemented solution, we maintain an interactable hand with good precision while simultaneously managing an acceptable computational load in the physics calculations.

3.3.4 EasyHandRig Template Implementation

To seamlessly integrate the concepts presented above into the VR environment, we have devised a template structure known as 'EasyHandRig'. This template simplifies the process for developers, enabling them to effortlessly incorporate functional hand tracking into their projects. The architectural components of this template are illustrated in Figure 3.11.

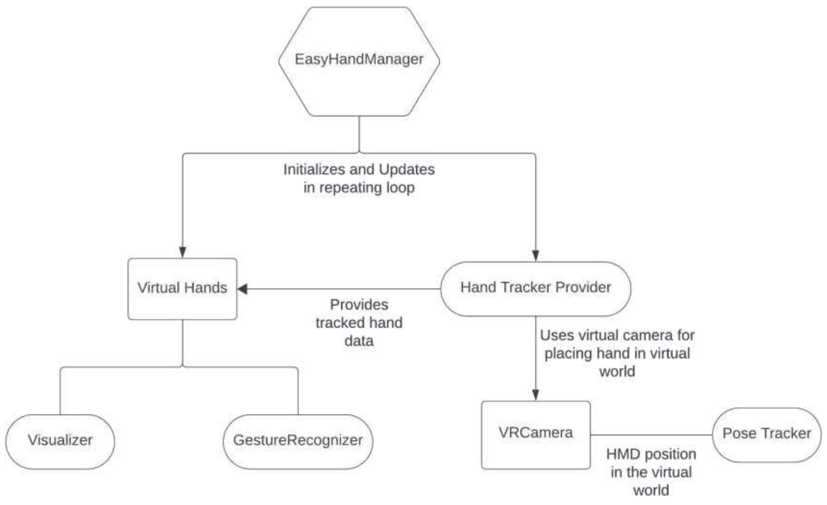

EasyHandManager
Initializes and Updates in repeating loop
Virtual Hands
Hand Tracker Provider
Provides tracked hand data
Uses virtual camera for placing hand in virtual world
Visualizer
GestureRecognizer
VRCamera
Pose Tracker
HMD position in the virtual world

Figure 3.11: Components for the 'EasyHandRig' template.

At the core of this template is the 'EasyHandManager' class, which serves as a central hub for developers to select their preferred hand tracking system (e.g., Quest or Ultraleap). This class is responsible for initializing the foundational tracking system, selecting the mapping joints as per the developer's specifications, and initializing the visualization and gesture detection subsystems.

The 'HandTrackingProvider' is responsible for providing the unified joint data for each rendered frame to both the visualizer and gesture recognizer. To render the virtual scene, we use a virtual camera that is equipped with a pose tracker that handles the translation of real-world head-mounted display (HMD) movements into corresponding virtual world transforms.

An example of the 'EasyHandRig' inside the Unity Engine is illustrated in Figure 3.12.

3.4 Multi-User Capabilities

To enable colocated VR scenarios, our system is designed to create multi-user VR experiences. To achieve this, we use Photon PUN (Photon Unity Networking)[8], a real-time cloud networking solution designed for multiplayer games and applications. Photon PUN operates on a client-host architecture, where clients communicate with a central server, which then relays messages to other connected clients. This approach

[8]Photon PUN: https://www.photonengine.com/pun (Accessed:2023-10-06)

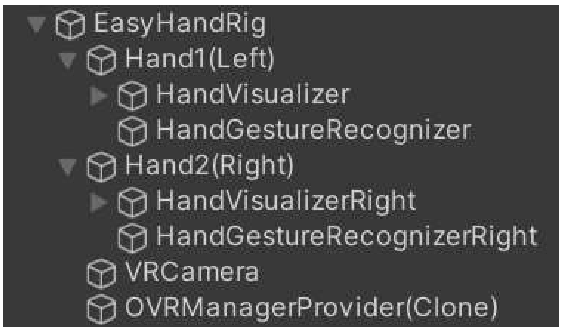


Figure 3.12: The 'EasyHandRig' inside the Unity engine. The 'EasyHandManger' component is placed on the parent 'EasyHandRig' game object. Each hand object has its own visualizer and gesture recognizer. The VR camera represents the user's HMD and is responsible for user positioning and rendering. In this example, Meta Quest hand tracking is used, which is why an OVRManager is created at runtime to obtain the low-level data of the tracking system for unification.

eliminates the need for direct communication between connected clients, simplifying the communication process when multiple clients are connected simultaneously.

Table 3.1: Overview over all synchronized that is sent to the Photon server and then broadcasted to all connected clients

Name	**Type**	**Size (Bytes)**	**Sync Rate**
Connect ID	Integer	4	Once
Rig Transform	Vector3 & Quaternion	28	60 Hz
Camera Transform	Vector3 & Quaternion	28	60 Hz
Hand Data	Hand	373++	60 Hz
Gestures	GestureEventArgs	9++	Once when detected

Synchronizing Data Photon automatically configures the cloud server to broadcast received data to all connected clients while also managing matchmaking and handling connection-related events, including connection losses. To minimize latency and optimize data handling, only essential data is sent to the server and subsequently broadcasted to all clients. An overview of the synchronized data is provided in Table 3.1. In cases where '++' follows the sent package size, it indicates that the sent data has a minimum size specified but may be larger due to the inclusion of dynamic data, such as strings with variable lengths.

Hand De-/Serialization As mentioned in Table 3.1, the synchronization of detected hand data over the network is a needed component of our system. Leveraging our unified data class for recognized hands, we can precisely define the data to be serialized. Given that connected clients require more than just joint positions, we also synchronize additional relevant information. Table 3.2 provides an overview of the elements needed for synchronization.

Table 3.2: Synchronized data for a serialized hand

Name	**Type**	**Size (Bytes)**
Left or Right Hand	Boolean	1
ID Length	Integer	4
ID	String	ID Length
Hand Transform	Vector3 & Quaternion	28
Tracking Method	Integer	4
21 Joint Indices	Integer	21 x 4
21 Joint Positions	Vector3	21 x 12

This results in a data set with a minimum size of 373 bytes, generated from the hand data class during each synchronization cycle. This data set is then transmitted to the clients and parsed back into the appropriate data class. Subsequently, it can be employed for visualization and interactions, as detailed in the previous sections.

The same process is applied to recognized gesture data. Alongside the recognized gesture itself, which occupies 4 bytes in size, we transmit additional information, including whether the gesture is performed by the left or right hand (1 byte) and the sender's ID (4 bytes for the ID length, with a custom size of bytes for the actual ID, since we don't know the length of the ID String). This information enables every client to definitely associate a gesture with a specific hand of a network user.

3.5 Distribution and Extensions

The framework has been developed as an extension for Unity3D, designed to be modular and easily accessible for developers to seamlessly integrate into both new and existing projects. It adheres to the structure of Unity's UPM (Unity Package Manager) workflow [109] and is hosted on the Git provider 'Bitbucket'[9]. This enables developers to effortlessly incorporate the package and all its dependencies through Unity's Package Manager using the provided Git URL. An overview of their integration is depicted in Figure 3.13 at the 'System Integration' part.

To streamline the integration process and minimize overhead, we have externalized base-tracking systems and additional features as separate packages. The core framework comprises the following essential components:

- **Core Source Code**: Containing fundamental code, including data structures, mapping, visualization, gesture recognition, and physics.

[9]EasyHand Repository: `https://bitbucket.org/Densen90/easyhand` (Accessed: 2023-10-17)

- **MediaPipe Tracking**: Handling the logic and integration of the MediaPipe system, enabling hand tracking from webcams (see chapter 5).
- **Networking**: Offering seamless network integration, facilitating multi-user capabilities, and serialization / deserialization of hand data.
- **Hand-Ownership Estimation**: Containing classes for estimating hand ownership using straightforward methods.

Should developers require additional functionality, custom plugins can be incorporated. A plugin is essentially an archive hosted on a web server containing the necessary logic implementations, external dependencies, packages, and the required assets and resources. The following plugins are available and provided by us:

- **LeapIntegration**: Allow hand tracking with the LeapMotion tracking controller.
- **QuestIntegration**: Allowing hand tracking with the Meta Quest or supported Meta HMDs.
- **ViveIntegration**: Allowing hand tracking with all compatible HMDs using HTC hand tracking technology.
- **MLHOAgentsIntegration**: Enhancing the hand ownership algorithm by integrating Unity3D's machine learning agents for assigning virtual hands (see section 6.2).
- **Colocation**: An additional UPM package that relies on the 'EasyHand' framework, facilitating the creation of colocated VR scenarios. This can be achieved using tracked hands, initial HMD placements, or AruCo markers (see section 4.2).

Archives for accessing these plugins, as well as the Colocation UPM package, are also hosted on servers of the git provider 'Bitbucket'[10].

3.6 Current State

The 'EasyHand' system currently meets all the requirements outlined in section 3.1.1. A high-level overview of the system workflow is provided in Figure 3.13.

The components denoted as 'System Integration' and 'VR Headset Tracking' are customizable and can be chosen to suit the user's specific hand tracking and VR system preferences. The core logic for a unified Visualization, Interaction, and Gesture Recognition system, as explained in the preceding sections, is designed to seamlessly interface with these interchangeable integrated systems. The resulting unified data can then be synchronized

[10]EasyHand Plugins Repository: `https://bitbucket.org/Densen90/eh_plugins` (Accessed: 2023-10-17)

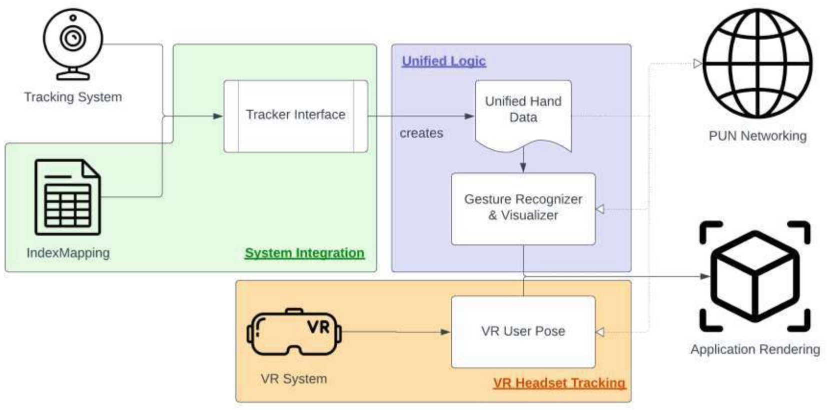


Figure 3.13: General overview of the flow of the 'EasyHand' system. 'System Integration' shows how base hand tracking systems are integrated. 'Unified Logic' is the core logic for Visualization, Interactions, and Gesture Recognition of the system. These parts can be synchronized over the network. 'VR Headset Tracking' is the integration of a VR headset to track the user's head movement and work in conjunction with tracked hands in the final application.

over the network to create a multi-user VR application. When there are multiple users present, the developer has the ability to use the Hand Ownership algorithm outlined in chapter 6 to discern which hand corresponds to each active user.

The 'EasyHand' framework is currently on version 1.1.8[11], implemented in Unity3D version 2022.3.5f1 and relies on Photon Unity Networking 2.30 for network functionality. At the time of writing this thesis, the framework supports the following hand tracking base systems:

- Meta Quest Hand Tracking with 'OculusIntegration' v.55.0
- LeapMotion Hand Tracking with Ultraleap Plugin v.6.10.0
- Vive Hand Tracking v.1.0.0
- MediaPipe with Hand Tracking Integration v.0.8.10, together with a Unity C# wrapper plugin v.0.10.3[12]

Future work on the system could focus on the automatic integration of current versions of underlying base hand tracking APIs. Currently, the system includes plugin archives

[11]EasyHand Repository: `https://bitbucket.org/Densen90/easyhand` (Accessed: 2023-10-17)
[12]MediaPipe Unity Plugin: `https://github.com/homuler/MediaPipeUnityPlugin` (Accessed: 2023-10-06)

hosted on a web server, each of which integrates a specific version of the base hand tracking API (e.g. OculusIntegration). Automatically referencing the officially released versions of these APIs would serve to reduce the file size of hosted archives. This approach would make it more convenient for developers to use the system and provide access to the latest features and performance updates of the tracking systems without the necessity of manually updating the plugin archives on the web server.

Additionally, the integration of hand tracking capabilities of the OpenXR standard could expand the utility of the 'EasyHand' system. This enhancement would enable compatibility with the universal XR platform, allowing for the seamless integration of multiple additional hand tracking systems, such as Microsoft Mixed Reality [32]. This expansion would facilitate uniform XR hand tracking within the system, further enhancing the user experience. Furthermore, it would open opportunities to incorporate networking capabilities into OpenXR, as discussed earlier. These advancements would contribute to making the 'EasyHand' system more widely accessible and user-friendly.

CHAPTER 4

Using Tracked Hands to Create Colocated VR

The following chapter will present our methods for synchronizing the coordinate systems the virtual worlds of two independent users using SLAM-tracked VR headsets. We will introduce three distinct approaches: one that involves the alignment of the HMDs, another where both systems track an ArUco marker, and a third where the HMDs are aligned by tracking the hands of the same user. Following this, a comparative experimental analysis will be performed to evaluate the calibration error, and we will discuss the optimal approach considering factors such as consistency, usability, ease of setup, and scalability.

4.1 Motivation

In a colocated VR setup, poses of all users within the same coordinate frame need to be known. To achieve this, virtual users must be placed in correspondence to each others locations concerning their position and view direction. Figure 4.1 illustrates this. In setups where external cameras are used, this approach is easier, as the external camera system establishes the same shared coordinate system for all users inside the tracking range of the system. Such systems can utilize, for example, the Lighthouse tracking system of the Vive [122] or external camera systems like OptiTrack for large physical environments [29]. However, these systems are intricate to set up, come with high costs and are limited in their mobility. For these reasons, we aim to have no dependency on these systems for the creation of colocation.

At the time of writing, head-mounted displays that use built-in visual SLAM techniques for head tracking are gaining popularity in the consumer VR market. These demonstrate the substantial advantage of not requiring an external camera setup for fast and precise 6DOF tracking. They can often allow virtual environments that are larger than those

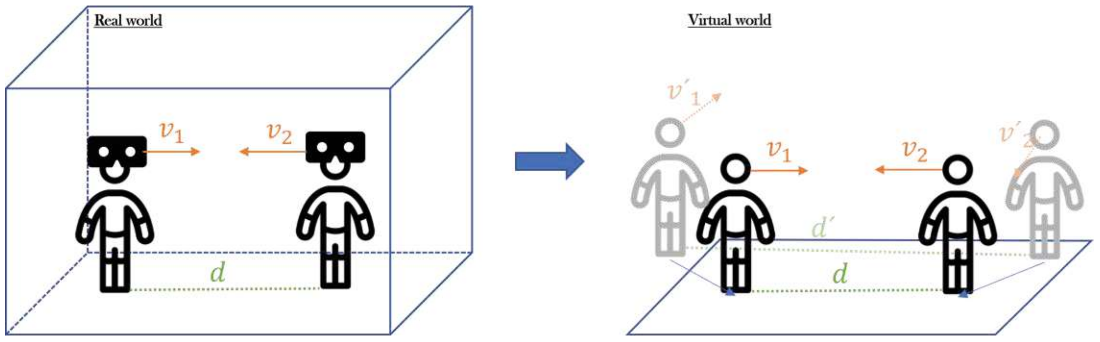


Figure 4.1: Sketch of requirements for a colocated scenario. *Left:* Two users standing in front of each other at a distance d and their view directions v_1 and v_2. *Right:* Their virtual representations are aligned to have the fitting distance and view direction of their real-world counterparts.

that similarly-priced external tracking camera installations can cover. The Meta Quest is an example of an HMD that uses SLAM and is available to consumers.

As a SLAM-tracked device, each Meta Quest creates an individual environmental tracking map. However, for colocated shared environments, synchronizing the virtual space for all users becomes a challenging task, given that the tracking map cannot be read out and copied to other devices. As we aim to avoid reliance on external tracking systems for map synchronization, given the constraints of most devices that can be integrated with the HMD, our focus is on developing methods that synchronize exclusively with the user's SLAM-tracked HMD — in this case, the Meta Quest headset.

We address this challenge by investigating three methods that initially calibrate SLAM-tracked HMDs within the same physical environment to create a shared colocated VR scenario:

- **Fixed-point calibration:** All colocated HMDs are initial placed at predefined positions within the physical environment.
- **Marker-based calibration:** A marker in the physical environment is tracked simultaneously by all client applications running on the user's HMDs.
- **Hand tracking-based calibration:** The hands of one of the colocated users are used as spatial anchors, simultaneously tracked by all client applications.

Although two of the investigated calibration methods - fixed-point calibration and marker-based calibration - have been used in previous research, to our knowledge, the hand tracking-based calibration method is entirely novel.

To summarize, we present the following contributions with our experiment:

- A new calibration method for shared colocated VR scenarios using SLAM-tracked HMDs with hand tracking. This method employs user hands as spatial calibration anchors, eliminating the need for additional infrastructure and demonstrating superior calibration accuracy.
- An experimental comparative evaluation of the accuracy of three colocation calibration methods.
- Analysis of limitations and future possibilities of the discussed calibration methods.

To achieve our overarching goal of creating colocated shared environments with hand tracking, where tracked hands can be obtained by multiple systems and assigned to the correct user, the method presented here for creating colocated shared environments for SLAM-tracked headsets with the help of hand tracking is the first necessary step.

4.2 Calibration Methods for Creating Colocated VR

In this section, we present the design and implementation of three different calibration methods that enable shared colocated VR scenarios. All discussed methods are designed to work with SLAM tracked headsets in general, are independent of external tracking systems and tested on the Meta Quest HMD. The details of our implementation of two previously published calibration methods - fixed-point calibration and marker-based calibration - are followed by the description of the design and implementation of our novel hand tracking-based method.

4.2.1 Fixed-Point Calibration

This calibration method is the most simple and straightforward of the three presented methods. To prepare the physical environment for calibration, a specific point is predefined and marked (for example, the center of the room). We call this point U_R. Then, a point in the virtual world U_V is manually set up that should correspond to U_R. After calibration, a user at position U_R in the physical space should have the position U_V in the virtual space. A set of distinct U_R and U_V positions is determined for each colocated user. Some applications use the same reference points for all users, calibrating user positions one after another [70]. We choose to set up a unique reference point pair for each user, enabling simultaneous calibration.

For the calibration, we position the HMD of each user on the floor at their corresponding U_R, rotated in the direction that is set with U_V. Then we manually start the calibration process to align the virtual users with their reference points.

Let U_{prev} be the virtual starting point of the user and U_V be the pose we want the user to be aligned with. We define the position and rotation of the virtual user with$\{p_{prev}, r_{prev}\} \in U_{prev}$ & $\{p_V, r_V\} \in U_V$. We then determine the amount we have to rotate the user (α) and the amount we have to move the user (Δp) with the following formula:

$$\begin{aligned}\Delta p &= p_V - p_{prev} \\ \alpha &= r_V * r_{prev}^{-1}\end{aligned} \tag{4.1}$$

The process can also be seen in Figure 4.2, where the location and orientation of the user's heads are aligned with their physical HMD counterparts.

In the setup used in our evaluation experiment described in section 4.3, the reference points of two users were placed one meter apart from each other, with the HMDs rotated to look at each other. However, reference points can be set at arbitrary distances, as long as their relative poses in the physical world correspond to those in the virtual environment.

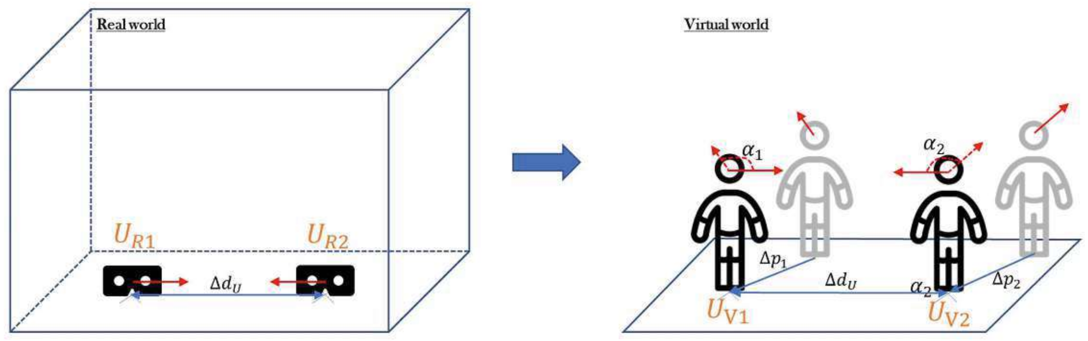


Figure 4.2: *Left:* The users' headsets are positioned on predefined locations in the real world. *Right:* Virtual users who are repositioned to U_V, which is the virtual representation of U_R. Distance Δd_U is the same in the real and virtual world. Red arrows represent the view direction of the user.

4.2.2 Marker-Based Calibration

Our implementation of the marker-based calibration method uses an ArUco marker placed on the floor in the tracking space as a spatial anchor for the colocated HMDs. ArUco markers are square images featuring a black border and an inner matrix represented by white dots denoting the marker's ID [84]. These markers are easily created and adequately unique for our use case, as only one marker is required for our setup. We used a ZED mini camera that we attached to Meta Quest to detect the ArUco marker. However, HMDs front-facing cameras could be used as well, as long as their video feed can be made accessible to the developer. Using the OpenCV framework, we calculate the position and rotation of the detected marker in camera space to recalculate the position and rotation of the camera in the coordinate frame associated with the marker.

This marker also has a reference representation in the virtual world. This representation is our virtual anchor from which we are relocating our users.

Compared to fixed-point calibration, where U_V is known as the location we want to be aligned with, in marker-based calibration, we have to calculate this pose after a marker was detected. Since we know the pose of the world space markers (M_V) in the virtual world, we can determine the location of the world space (U_V) to which we want to align our user. The marker recognition software provides us with the marker location in the user's camera space. This can be inversed to determine the user's location in the marker space (we define this with U_M). We now are able to determine U_V with $U_V = M_V + U_M$. Since we now know U_V, we can relocate the virtual user to this location using Equation 4.1. An illustration of the calibration process can be found in Figure 4.3.

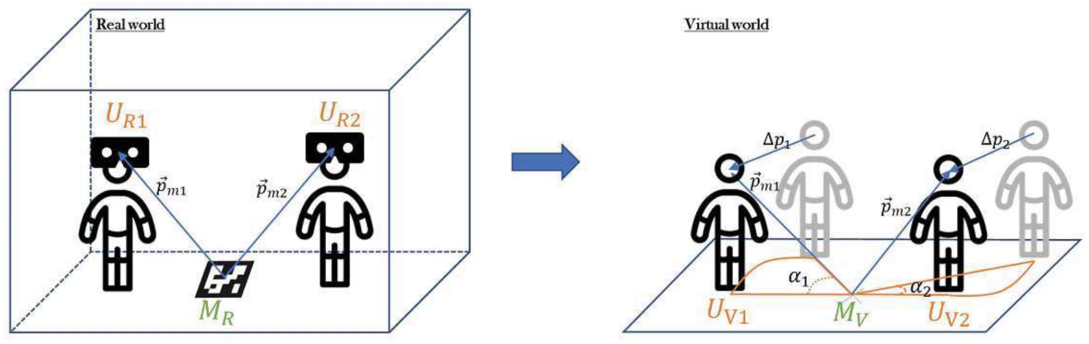


Figure 4.3: *Left:* The users standing in the real world detecting an ArUco marker. *Right:* Virtual users who are relocated depending on the detected marker. The virtual user is moved by Δp and rotated by α to get to U_V, which is the position of the user $\vec{p_m}$ and rotation $\vec{r_m}$ in the marker space.

This calibration can be used for each user independently or simultaneously. The process is either triggered by a user interaction (e.g. pressing a button on a controller) or, in our case, from an admin computer. This way, it can be controlled and easily redone during the experiment, as long as the marker is in sight of the AR camera. Figure 4.9 shows this setup in the real world.

4.2.3 Hand Tracking-Based Calibration

Our novel calibration method based on hand tracking of colocated users requires a hand tracking-capable device. We worked with Meta Quest in our implementation; however an additional sensor such as a LeapMotion device attached to the HMD could be used.

To see whether it makes a difference in accuracy or effort, we implemented two recalibration methods with hand tracking. The first one uses one tracked hand (here the right one) to relocate the user according to the tracked pose. The second one relocates the user by tracking both hands and calculating a mean pose to which the user is reoriented.

During the calibration process, one user's hands are held behind their back to prevent the headsets from accidentally tracking these hands. The other user is then holding their right hand (respectively both hands) in front of both headsets, so both track it. If this is the case, the colocation process can be triggered by a user interaction (e.g. pressing a button). The calculations for that are found in Equation 4.2. In our setup, an admin computer is triggering the colocation process. However, in future implementations, this could also be done by one of the users.

Then one user sends the position and rotation of their hand anchor to the other user as a reference point. For one hand, this is the tracked hand pose; for both hands, it is a mean pose of both tracked hands. The receiving user then reorients his virtual hand to match the received pose. The whole user is then reoriented by the difference between its own and the received hand pose.

$$\begin{gathered}\Delta p = p_{refHand} - p_h \\ p_{user} = p_{user} + \Delta p \\ \Delta r = r_{refHand} * r_{ownHand}^{-1} \\ r_{user} = r_{user} * \Delta r\end{gathered} \tag{4.2}$$

In this method Δp is the amount we move the user's position and Δr the amount we rotate the user to get them to the location U_V. Figure 4.4 illustrates the reorientation process. U_V is the location we want to set the users to in the virtual world.

From now on, the users are colocated. Depending on the HMDs tracking accuracy, this reorientation can be redone every time the headsets' drift gets too big. However, such an increase in drift did not occur in the experiments carried out in this study. Since it does not require any preparation in the real world, this can be done anytime, anywhere during application. The only requirement is that the reference hand is visible for both users. Compared to the other methods, not all users are repositioned to a new virtual location. Since one user's hand is sent as a reference to other users, this user does not need to get reoriented because other users are relocated to match the reference position.

Variant based on the tracking of two hands For the calibration method where two hands are used, a mean point of both tracked hands is used. This point is the mid point between p_l and p_r which are the virtual points for the detected left and right hand and calculated as:

$$p_M{=}p_r + \frac{p_l - p_r}{2} \tag{4.3}$$

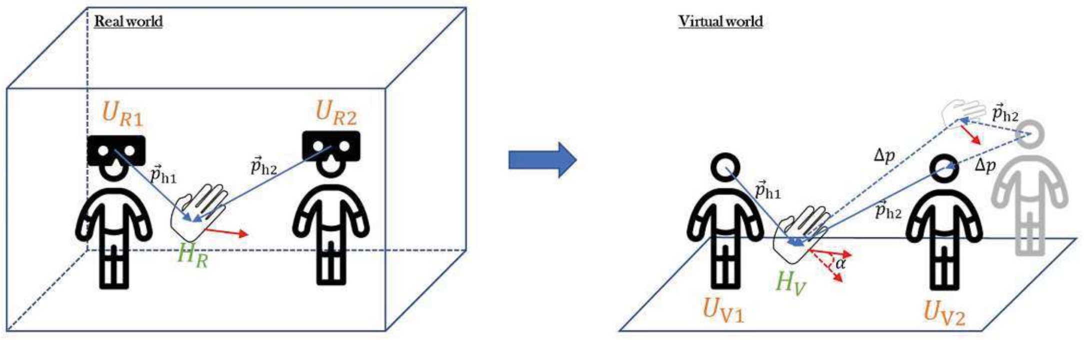


Figure 4.4: *Left:* The users standing in the real world detecting the same hand. *Right:* User gets relocated by difference Δp. α is the difference in rotation between the tracked hand and the received reference hand. Rotation is visualized by red arrows. Compared to other methods, only the other user gets relocated.

This formula is used to get the mean of the position and the rotation, that are then used as a single reference point when recalibrating the user, as explained in section 4.2.3.

4.3 Usability Evaluation Experiment

The goal of our experiment is to evaluate the usability of each of four presented calibration methods in terms of calibration accuracy, difficulty of setup and the need of additional hardware. Our four calibration methods are clearly distinct with respect to the calibration effort and the need for additional hardware and software. While this experiment comprises a first technical evaluation, we plan to conduct a user-centered evaluation with multiple participants as part of our future work[1].

4.3.1 Experimental Design and Evaluation

The evaluation was performed with two HMDs (Meta Quest) colocated within the same room and calibrated with each of the described methods. To estimate the precision of the calibration, we used ground-truth tracking data obtained with an externally mounted Lighthouse 2.0 tracking system with two sensor stations. A HTC Vive tracker was attached to each HMD (pictured in Figure 4.5), allowing the ground-truth distance between both HMDs $d_{GT}(t)$ in the frame t to be calculated as the distance between two trackers, adjusted by offset between the centre of the tracker and the centre of the HMD.

[1]Due to the COVID-19 pandemic, an extensive usability study with multiple users was not feasible at this stage.

The calibrated distance between the HMDs $d_C(t)$ in the frame t was calculated as the difference between their positions in the virtual scene. The difference between $d_{GT}(t)$ and $d_C(t)$ provides the final distance error in the frame t as described by Equation 4.4.

$$\delta(t)=|d_{GT}(t) - d_C(t)| \tag{4.4}$$

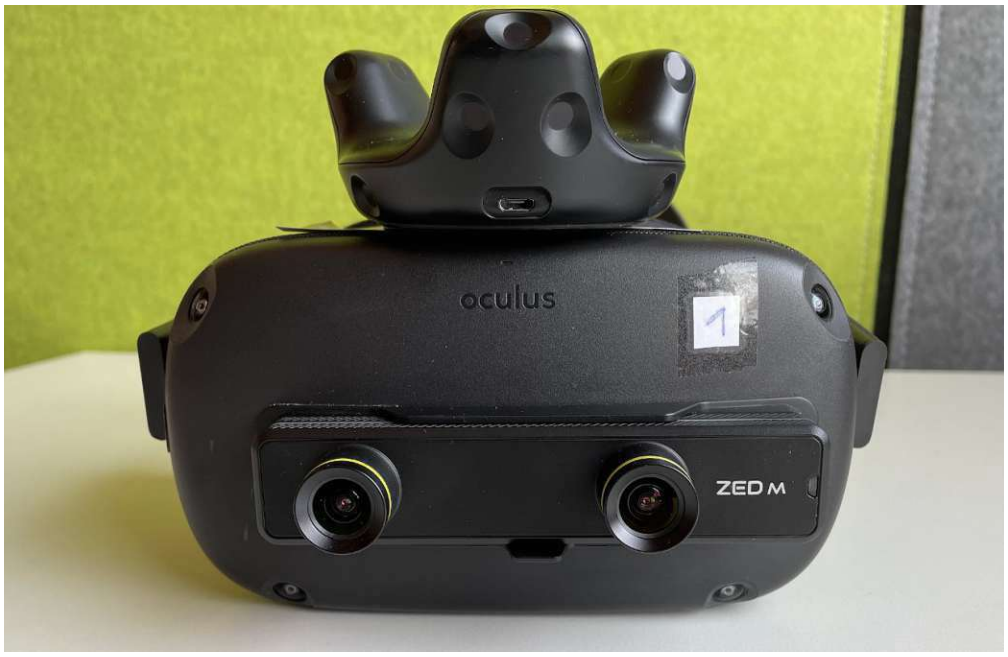


Figure 4.5: Meta Quest with an attached Vive Tracker and ZED-Mini camera used in the evaluation.

It is worth noting that $\delta(t)$ contains possible contributions due to imprecision or drift of the in-built SLAM-based tracking of Meta Quest as well as the influence of tracking errors inherent to the Lighthouse tracking system used as a ground-truth reference. The tracking accuracy of a Vive tracker delivered by the Lighthouse system has been shown to be in the millimeter range with high reproducibility of position measurements, making it viable for collecting ground-truth measurements [4][7]. In comparison, the tracking accuracy of Meta Quest was measured at a level below 1 cm under good lighting conditions [88]. These results motivate our use of the Lighthouse system as the source of ground-truth tracking data, as we believe its accuracy is sufficient to allow the comparison of the investigated calibration methods. To minimize the impact of these tracking errors, we calculate $\delta(t)$ for a number of frames after each calibration.

4.3.2 Pilot Evaluation

We collected pilot evaluation data, performing the calibration five times for each calibration method and calculating the distance error $\delta(t)$ in N=1000 consecutive frames after

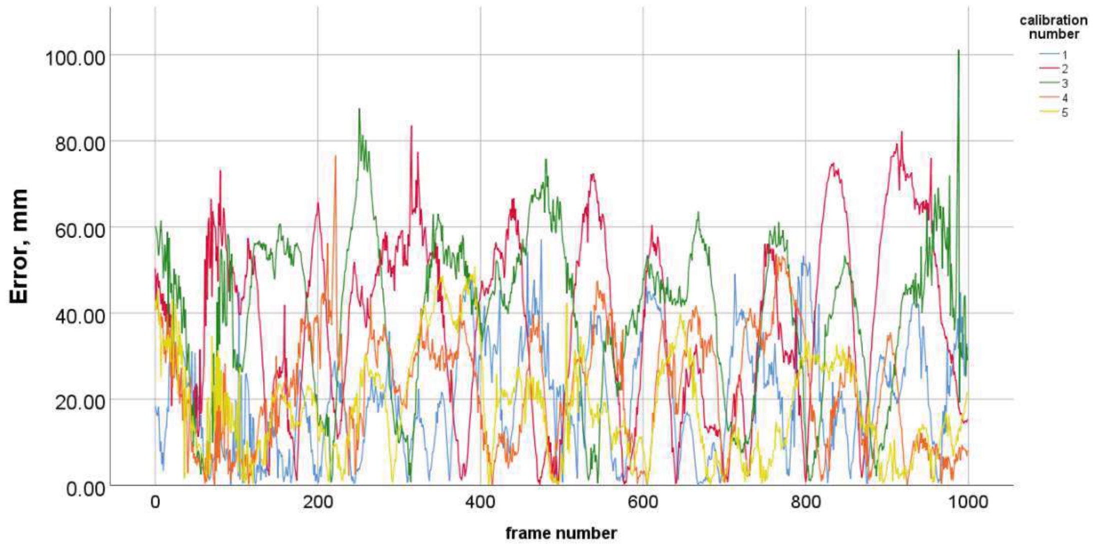


Figure 4.6: Time distribution of the distance error, shown on the example of a dataset from fixed-point calibration.

each calibration. These pilot recordings showed that, for each calibration method, the calculated error of the distance between two HMDs $\delta(t)$ is not correlated with time. An example of time distribution of the distance error can be seen in Figure 4.6. This example shows the tracking error over time during the 1000 frames recorded. The variance could be explained by inaccurate positioning in the fixed-point calibration, but this will be discussed again later. Furthermore, for each method, the median distance error was different for different calibration events. This fact was established with a Friedman's ANOVA repeated-measures non-parametric test for each calibration method, since all error distributions were not normal [25][26][27]. The box-plots of distance error obtained from the pilot recording are presented in Figure 4.7.

The box-plots indicate a considerable number of outliers in the distance error distributions. We are inclined to think that these outliers are the result of tracking inaccuracy in individual frames during recording sessions. However, our method does not allow to distinguish between contributions of possible inaccuracy of the SLAM-based tracking of Meta Quest or the Lighthouse system.

The pilot analysis shows that neither of the investigated methods provides consistent calibration performance across different calibration attempts. For the fixed-point method, the inconsistency in the calibration results can be readily explained by the inconsistency of precision with which the users place the HMDs on predefined calibration positions. For the remaining methods, calibration success directly depends on the precision of the various tracking data (of marker or users' hands) in the frame where the calibration took place. This dependency on the precision of the tracking data used in a calibration method could be addressed in future modification to the calibration methods, however, potentially at the cost of additional setup effort on the part of the users. We will discuss potential mitigation strategies in section 4.4. To address the impact of the varied imprecision of every individual calibration, as well as to eliminate the impact of outliers present in the measurement of the distance error, we chose to perform each type of calibration multiple times and to record the distance error in a large number of frames after each calibration. Afterwards, we compare median distance errors resulting from each calibration. We further use the term calibration error to refer to the median distance error calculated from the recorded distance data after each calibration.

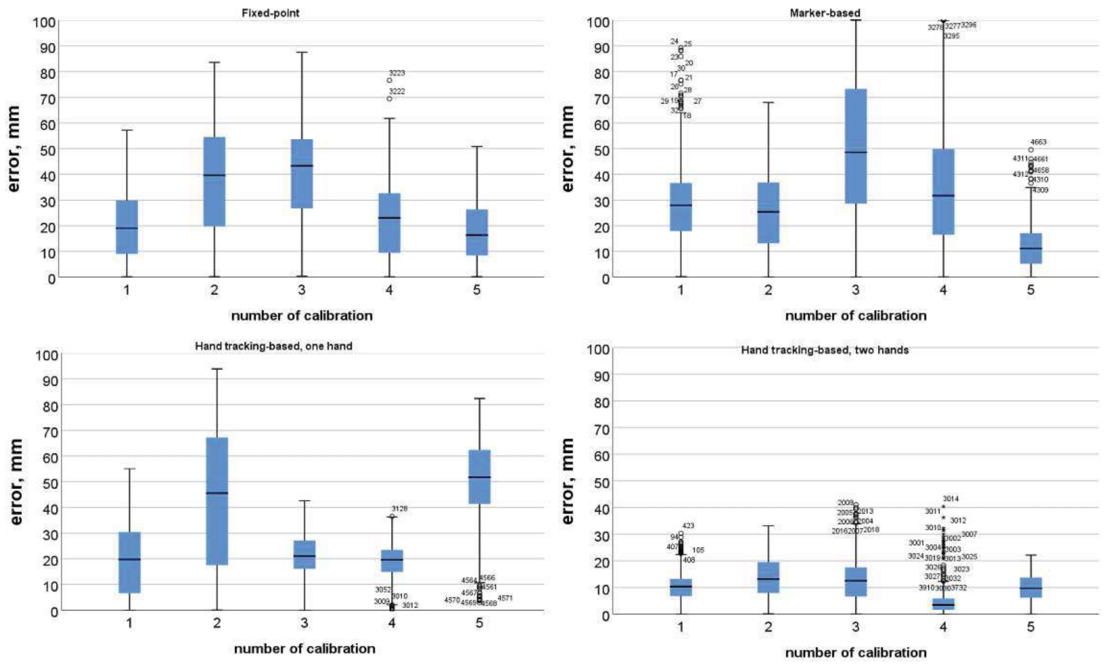


Figure 4.7: Distance error box-plots of pilot recordings.

4.3.3 Setup and Procedure

A distributed VR application run in the experiment is developed with Unity 3D (v.2019.4.3), with the networking layer built on the basis of the Photon Unity Plugin (PUN). The networking layer insured the synchronization of poses obtained with input tracking data and the simultaneous execution of experimental commands on all machines. Meta Integration asset for Unity 3D was used as an API layer providing tracked head and hand poses of Meta Quest to each Unity3D client application. However, the rendering of tracked user hands was achieved with the help of the 'EasyHand' framework that was presented in chapter 3, providing a universal layer for collecting and distributing hand tracking data obtained from any input source. In the experiment, both client applications running on Meta Quest were connected to a server, also running an administrative client issuing experimental commands. All poses of VR-immersed users as well as calibration-specific tracking data (of the tracked marker or tracked hands) are visible on the administrative client. The main command issued by the administrative client triggers the calibration procedure for a selected calibration method. A diagram illustrating the communication flow between the administrative client and Meta Quest clients is presented in Figure 4.8.

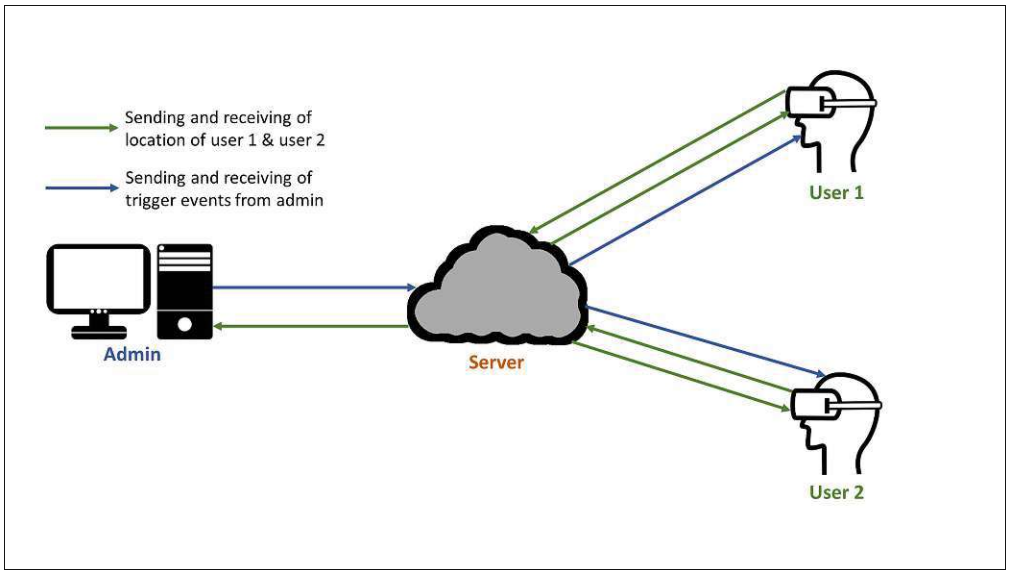

Sending and receiving of location of user 1 & user 2
Sending and receiving of trigger events from admin
User 1
Admin
Server
User 2

Figure 4.8: Network communication between admin computer and VR users.

The evaluation data was collected with the following experimental procedure:

1. Start the administrative client, which is also the master client (host) in the PUN distribution pipeline, to open the network connection.
2. Connect both HMDs with respective connected Vive trackers.
3. Ensure correct synchronization and assignments of HMDs and Vive trackers in the administration client.
4. Collect data following the procedure detailed in Section section 4.3.1.

To enable marker-based calibration methods, we attached a ZED-Mini camera to each HMD as demonstrated in Figure 4.5. Figure 4.9 illustrates our experimental setup on a room-scale. An ArUco marker was positioned on the floor in the middle of the room. Two markers for one headset each were placed on the floor at a distance of one meter for fixed-point calibration. When users connected to the same distributed application, they were able to see each other in the same virtual environment, although their relative positions did not coincide with those in the real room before calibration. After the calibration procedure was triggered from the administrative client, users' relative positions in the virtual environments were moved to coincide to their relative positions in the physical environment and recording of their poses started. Although the evaluation was conducted with two colocated users, the calibration procedure accommodates an arbitrary number of users.

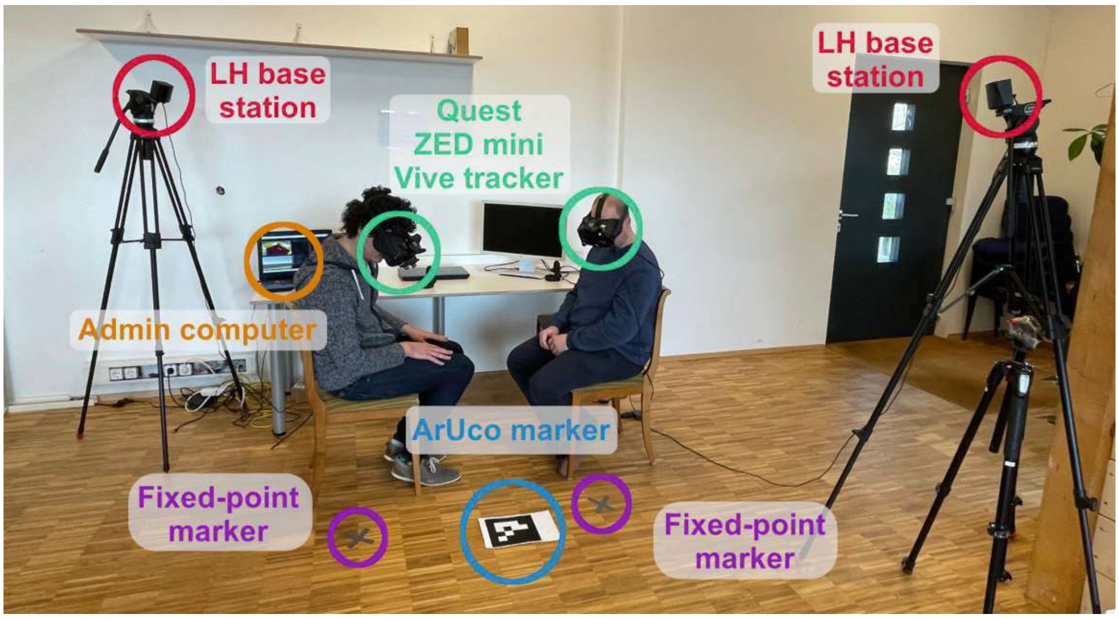


Figure 4.9: Exemplary experiment setup for marker-based calibration method.

4.4 Results and Discussion

The following section will present the results of the analysis of the aforementioned evaluation. After stating the calibration error of the several methods we will discuss these results and elaborate on further important properties of the evaluated calibration methods for colocated SLAM-tracked HMDs.

4.4.1 Calibration Error Outcome

Data used in the evaluation was collected following the approach described in section 4.3.1. For each method, the calibration was performed 25 times, resetting the application each time and attempting to perform the same head movements each time; the error between the ground-truth distance between the HMDs and the distance derived from the calibrated positions was recorded during 1000 frames after each calibration. We then calculated the median error for each dataset of 1000 error values, obtaining four datasets of median errors with 25 entries in each. The box-plots of these four datasets are presented in Figure 4.10. The median errors proved to be normally distributed in the Shapiro-Wilk test [102] after two outliers had been removed ($p = 0.055$ for the fixed point method, $p = .102$ for the marker-based method, $p = 0.021$ for the method based on one hand hand tracking, $p = 0.066$ for the method based on two hands hand tracking). The removed outliers can be seen in Figure 4.10, one in the marked-based method and one in the hand tracking-based method with two hands. We then used one-way ANOVA [85] to compare the distributions means. The analysis was performed with IBM SPSS Statistics.

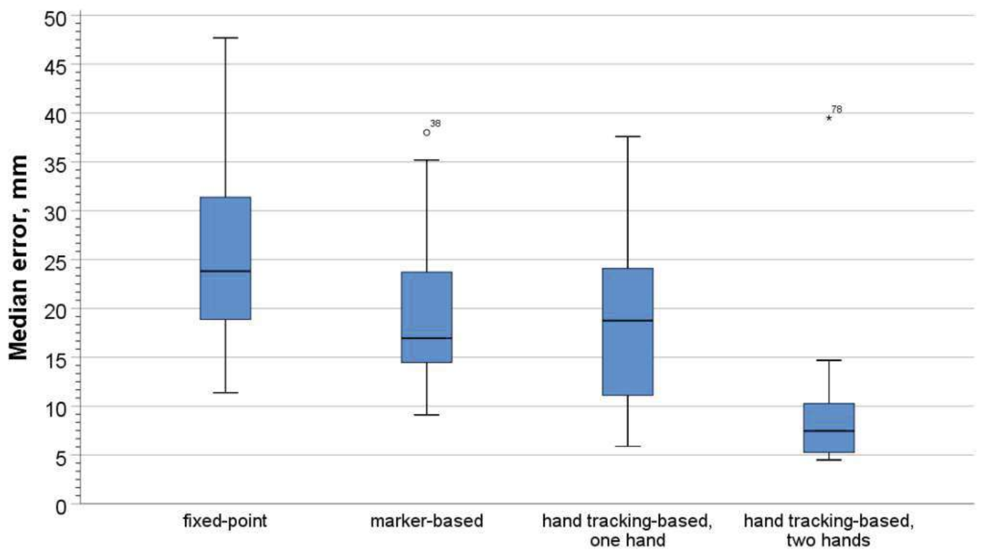


Figure 4.10: Box-plots of median distance error for four calibration types.

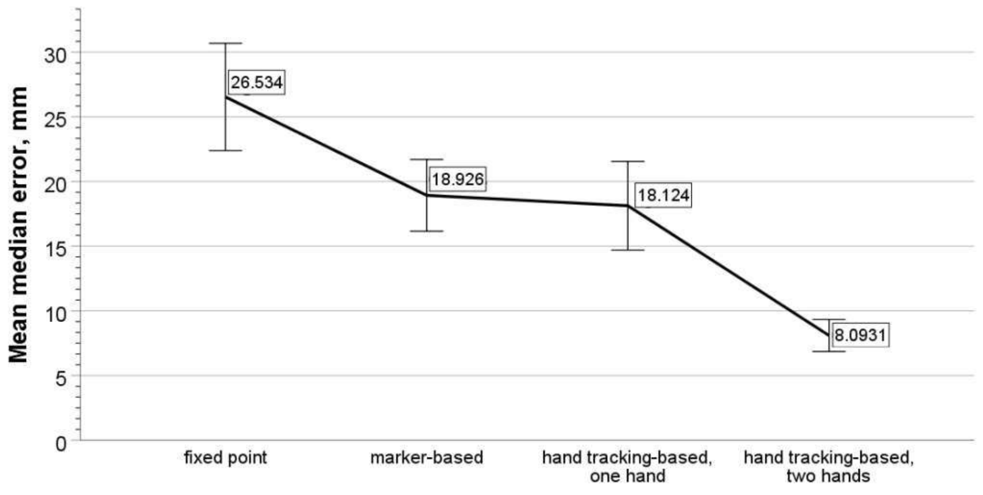


Figure 4.11: Mean values of calibration error (median distance error) for each calibration method.

Levene's test [60] showed a violation of the homogeneity of variance (p < .001). Therefore, we used Welch's test [123] for the main analysis and the Games-Powel test for post-hoc comparisons. The median error differed significantly with different calibration methods

(Welch's F(3, 46.765) = 40.965, p < .001). The resulting plot of mean values is presented in Figure 4.11. Post-hoc analysis revealed that the median error was significantly larger for the fixed-point method than for all other methods. The median error of the hand tracking-based method with two hands was significantly smaller compared to all other methods. Details of post hoc comparisons are summarized in Table 4.1. Mean differences are reported as significant at the 0.05 level.

(i) method	(j) method	mean diff. (i - j)	std. error	sig.	95% CI	
					lower bound	upper bound
fixed-point	**marker**	7.60789	2.49586	**.02**	.9291	14.2867
	one hand	8.41017	2.68948	**.016**	1.2433	15.5770
	two hands	18.44063	2.16391	**< .001**	12.5353	24.3460
marker	**fixed-point**	-7.60789	2.49586	**.02**	-14.2867	-.9291
	one hand	.80227	2.20473	.983	-5.0771	6.6817
	two hands	10.83273	1.51988	**< .001**	6.7132	14.9523
one hand	**fixed-point**	-8.41017	2.68948	**.016**	-15.5770	-1.2433
	marker	-.80227	2.20473	.983	-6.6817	5.0771
	two hands	10.03046	1.82044	**< .001**	5.0816	14.9793
two hands	**fixed-point**	-18.44063	2.16391	**< .001**	-24.3460	-12.5353
	marker	-10.83273	1.51988	**< .001**	-14.9523	-6.7132
	one hand	-10.03046	1.82044	**< .001**	-14.9793	-5.0816

Table 4.1: Results of post-hoc pairwise comparisons with Games-Powel test.

The results of our evaluation show that the fixed-point calibration method proved to have the largest median calibration error, whereas the hand tracking-based method provided the most accurate calibration when the variant based on the tracking of two user hands was used. The accuracy results of the marker-based method and hand tracking-based method using one user hand are comparable, with their accuracy being higher than that of the fixed-point method but lower than those of the hand tracking-based method using two hands.

The observed higher accuracy in methods using hand detection and ArUco markers, compared to the fixed-point method, can be attributed to the inherent limitations of the fixed-point method. The fixed-point method relies on manual placement of the HMD and lacks the automated detection employed by the other methods. Consequently, the introduction of human error is a significant factor in this approach. In contrast, hand recognition utilizes multiple cameras on the HMD to precisely determine the position and rotation of hands in a three-dimensional space, ensuring a high level of accuracy in placement. Notably, the method involving two hands exhibits the smallest error. This can be attributed to both the high tracking precision afforded by the hand-tracking technology and the use of multiple anchors in space. The presence of two hands allows for error compensation, wherein any potential inaccuracies in the position of one hand can be mitigated by the information from the other hand. This is in line with the findings of McGill et al., who demonstrated that employing two fixed points can yield improved accuracy [70].

4.4.2 Consistency and Potential for Improvement

Consistency of the calibration result describes the extent to which each calibration method delivers similar calibration accuracy when the user performs identical actions to calibrate the colocated HMDs. In the pilot evaluation stage, we discovered that the median distance error measured after each calibration proved to be different for all the methods evaluated. Data of 25 calibrations for each discussed method allows us to have a more detailed look at the method's calibration consistency.

The fixed-point calibration method showed the highest variability among the four evaluated methods, demonstrated by the largest range of median error values and their interquartile range (Figure 4.10). This result is somewhat expected, given that each user needs to manually place their HMD at the marked spot to calibrate for a colocated scenario. It is hardly possible for users to achieve placement accuracy in the sub-cm range. The accuracy and possibly consistency of the fixed-point method could be improved by a two-point calibration procedure suggested by McGill et al. [70].

For the marker-based calibration method, the accuracy of each individual calibration is contingent on the accuracy of marker tracking in the frame where the calibration takes place. Compared to fixed-point and hand tracking-based calibration with one hand, this method shows a smaller interquartile range and span of median errors, indicating a better consistency than these two methods. Since the tracking process uses RGB images, its accuracy can be highly dependent on lighting conditions and varied in unstable lighting. A possible solution to mitigate the shortcomings of marker tracking is to collect marker pose data for several frames and use the averaged pose value in the calibration procedure. However, balanced values of frames need to be found since users would need to be very still during the marker pose collection time. Alternatively, a larger number of markers could be used in the calibration procedure, with the mean camera pose being calculated with the tracking data of all markers (similarly to the multi-marker tracking method used by Podkosova et al. [91]).

For calibration based on hand tracking using only one hand, the interquartile range and median error range are comparable to the fixed-point calibration range, showing much larger variability in distance error data compared to the setup when two tracked hands are used in the calibration process. This stark difference in the variability of error between the variants based on tracking of one hand and two hands might be an indication of the advantage of using multiple spatial anchors in the calibration process.

The hand tracking-based calibration method demonstrated the strongest consistency when two tracked hands were used in the calibration process. The increased consistency compared to the other tested methods is reflected in the much more compact span of the median error values and their interquartile range (Figure 4.10). According to our evaluation, the greater accuracy of this method combined with the clearly better consistency and ease of execution on the part of the users makes it the best method for calibrating two-user colocated scenarios.

4.4.3 Ease of Setup

The fixed-point calibration method does not require any additional hardware. It also means that no additional software or plug-ins are required for developers, making this method usable by a wide range of HMDs. However, the execution of the fixed-point calibration requires certain involvement on the part of users as they need to take place (or place their HMDs) at predefined locations as accurately as possible. Moreover, if recalibration is necessary during the application runtime, users would have to remove their HMDs to ensure that their positions in the tracking space are accurate.

Marker-based calibration might require additional hardware and software, depending on whether the HMD has integrated cameras that can be accessed to enable marker tracking or whether an external camera needs to be used, as in our evaluation. The use of marker tracking itself requires additional implementation. In addition, a marker must be positioned in a fixed location, which restricts mobility. For users, however, the execution of marker-based calibration does not present any difficulty since users only need to position themselves in a way that allows the calibration marker to be seen in the camera image. When recalibrating, users need to return to the marker, making calibration and recalibration dependent on the location of the real-world marker. The calibration process can be made even easier for users if continuous tracking is used and markers are attached directly to each user, as in the work of DeFanti et al. [15].

The hand tracking-based calibration method has similar hardware requirements as the marker-based calibration. Either an HMD with integrated hand tracking (for example, Meta Quest) or an external sensor (for example, LeapMotion) is needed. For both cases, the developer needs to implement hand tracking detection (i.e., use the tracking systems SDK). Since integrated hand tracking is becoming more ubiquitous, this calibration method can be used on more and more HMDs without requiring additional hardware. For preparation and calibration effort, this method is shown to be the least demanding among the evaluated methods. Neither a physical marker nor a fixed location has to be set in the real world. The only requirement is that the hands of one user must be simultaneously visible to both users. For recalibration, the users do not need to return to a specific spot in the real world. Still, they have to be near each other enough so that the reference hand is visible to both tracking systems.

4.4.4 Scalability

In our evaluation, two users were colocated in the same physical environment. However, each of the tested methods is designed to work for an arbitrary amount of users. In the following, we briefly discuss how the applicability of each method extends to larger amounts of colocated users. A future comprehensive user test can further verify this.

The fixed-point calibration method can be easily extended to accommodate any amount of users. A corresponding number of marked positions in the physical environment and their counterpart target positions in the virtual environment need to be prepared to extend the method. Although such preparations would require additional involvement of an application developer, the calibration difficulty for users will remain unchanged.

For the marker-based calibration method, there might be an upper limit on the amount of participating users since HMDs (or cameras attached to them) of all users would need to track the calibration marker simultaneously (in the case where a simultaneous calibration is required). However, this limit would be rather large - it should be possible for up to ten users to stand in a circle so that the calibration marker can be tracked in all client applications. Such a limit on the amount of user HMDs that can be calibrated simultaneously would most probably be larger than the number of users that can be physically colocated in a regularly sized tracking room. For larger tracking spaces that many users share, several markers with known offsets could be arranged to calibrate sub-groups of users. Otherwise, it would also be possible to rely on the low drift of the SLAM-tracked HMDs and calibrate the users one after the other. In this case, the number of users would also be limited by the size of the physical environment.

As with the marker-based calibration method, the hand tracking-based method could have an upper limit on the amount of participating users. For a successful calibration, the reference hand needs to be tracked simultaneously in all client applications. The range of hand tracking limits the number of possible user positions from which the reference hand can be tracked. The exact scale of this limitation needs to be examined in future work. For a larger number of users, the hand-tracking-based calibration could be separated into multiple calibration steps. Users' poses could be calibrated to the same reference hand one after another until all users are correctly colocated in the virtual environment.

4.4.5 Exploring Colocation in a Different VR Scenarios

The applicability of all four discussed calibration methods extends to various VR scenarios. The most common use cases relate to room-scale environments, where users stand and navigate the virtual space by walking (see Figure 4.12).

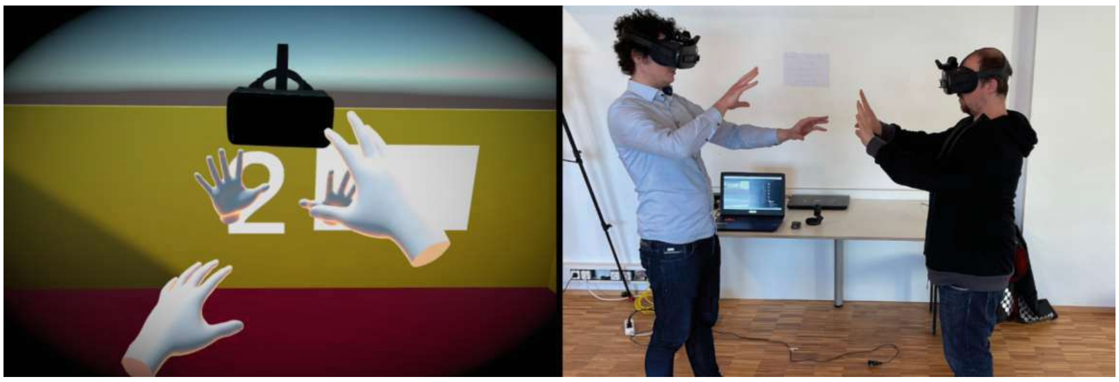


Figure 4.12: Two colocated users standing in front of each other with hand tracking enabled.

In future investigations, it would be worthwhile to delve deeper into room-size constraints and drift issues that may arise during walking, with a focus on determining the optimal frequency for recalibration.

But the colocation is not limited to room-scale scenarios. Seated colocated VR experiences enabled by SLAM-tracked HMDs also require calibration and environment synchronization. Colocation in seated VR can be used in a number of scenarios. For example, it can enable a meeting scenario where several participants are sitting at the same table. A calibration method will ensure that the virtual environment and all users are synchronized, allowing them to view and interact with the same 3D model or visualized data. It can also prevent collisions and enable the use of haptic elements or physical props in the environment.

Likewise, the discussed calibration methods can be used for seated collaborative VR experiences in CAVE environments [58], where users are placed in physical seats aligned with virtual seats. Currently, alignment is ensured manually by measuring the poses of the physical seats in the CAVE space. This alignment can be automated using a calibration procedure with an AR marker or hand tracking for more flexible scenarios. Figure 4.13 shows two seated users after being colocated interacting with their hands.

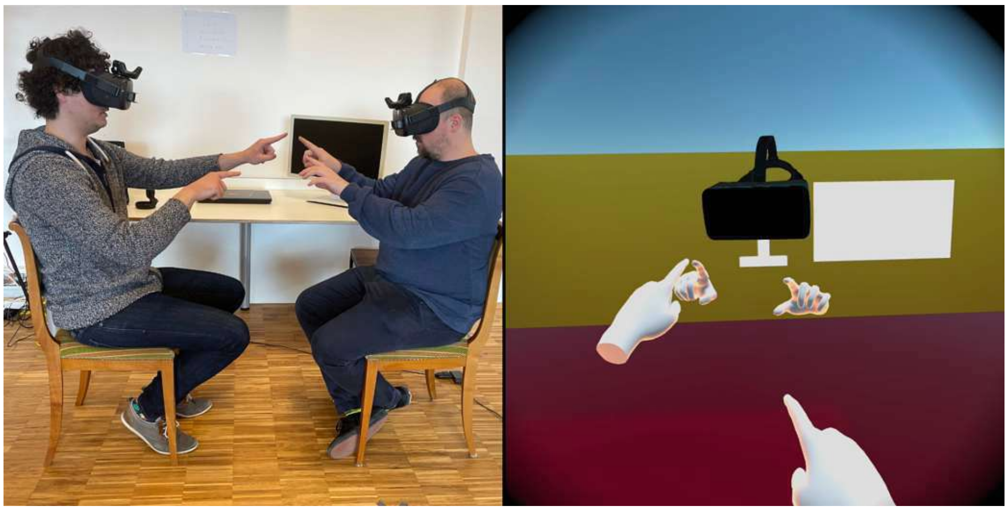


Figure 4.13: Two seated colocated users using their hands.

4.4.6 Applicability and Future of Hand Tracking-Based Calibration

Currently, Meta Quest is not designed to be used in colocated scenarios. This argument can be supported by the absence of access to the internal tracking map, which makes colocation calibration necessary in the first place. Its hand tracking-enabled interaction input is not designed to be used in colocated scenarios either, the possibility of tracking hands of users not wearing the Meta Quest device itself clearly being an artifact. It is this artifact that allowed us to use hand tracking for colocation calibration.

It is conceivable that, in the future, neural net training used to enable hand tracking on HMDs with forward-facing cameras would be implemented in a way that prevents the hands of other users from being tracked (for example, by taking arm poses into account during the training stage). In this case, direct use of hand tracking for colocation calibration would not be possible.

However, hand tracking capabilities of HMDs equipped with frontal cameras can be extended to track hands of other users deliberately, even if colocated users are relatively far away in the common walkable area. Such extension could be helpful in providing hand pose (and possibly derived full-body pose) estimations in situations where a user does not keep their hands in front of their head (for example, when the user's arms are kept down, alongside the body). If such augmentations to hand tracking are developed, they can prove beneficial for colocation calibration, potentially proving increased accuracy. Mutual tracking is a promising direction of future work on colocated VR environments overall.

4.5 Conclusion

The research presented in this chapter investigated three calibration methods that enable shared colocated VR scenarios for SLAM-tracked HMDs. We implemented and experimentally evaluated fixed-point calibration, marker-based calibration, and our novel calibration method that uses hand tracking data of colocated users as spatial anchors.

Our experimental evaluation showed that hand tracking-based calibration using two user hands as anchors achieved the highest consistent accuracy compared to fixed-point and marker-based calibration. Not requiring any internal infrastructure and being easy to execute at any time in a colocated scenario, our hand tracking-based calibration method proved to be very advantageous.

With the current trend of hand tracking being adopted by HMD manufacturers, this calibration method provides great potential for a wide range of VR solutions. Since the setup is easy to use for end-users, we hope that this encourages developers to implement colocation into end-user VR applications further. In future applications, it would be interesting to extend the methods to more than two users. A limitation of concurrent users and the impact of interference of all users and VR systems could be examined. A tracking system designed to track users' hands mutually could improve the user experience in colocated scenarios and be promising research. With this approach, the impact of view direction when tracking hands on calibration results is a promising topic of investigation.

After successfully establishing a colocated scenario using the presented method and demonstrating its applicability, the next step involves facilitating interaction among colocated users. To achieve this, it becomes essential to track more than two hands and enable these hands to interact within a three-dimensional virtual environment. This aspect will be explored in detail in the upcoming chapter.

CHAPTER 5

Evaluate and Improve an RGB-Based Hand Tracking Solution for Colocated VR Usage

In this chapter, we will provide a method to enhance the existing RGB hand tracking capabilities of the MediaPipe framework [64][130] by extending the 2.5-dimensional tracking of keypoints into three-dimensional space. This process involves a three-step approach: first, detecting the hand itself; second, estimating the real-world hand size of the user; and third, utilizing this information to calculate the three-dimensional world space of the hand.

Subsequently, in the evaluation section, we will assess the placement accuracy of this algorithm and compare it with state-of-the-art hand tracking systems such as Meta Quest and LeapMotion. This comparative analysis will be conducted in both static and dynamic environments, as well as in real usage scenarios. We will determine and compare the tracking ranges of all systems. Finally, our findings will be summarized in a conclusion where we discuss the applicability and usability of the proposed algorithm.

5.1 Motivation

In chapter 4 we presented methods to create colocated VR scenarios with the help of hand tracking. Ensuring consistent tracking of all users' hands within the shared workspace is crucial for enabling reliable interactions over a wide tracking range. To achieve reliable hand tracking for all colocated users, we propose a method that takes advantage of the tracking system's capability to track more than two hands and operates within an extended range of distances, and furthermore accurately position the hands in a three-dimensional space, thereby facilitating colocated hand interactions; this way, each tracking device which is worn by a user can provide tracking input not only for this user's virtual hands but also for virtual hands of colocated others. This idea is presented in Figure 5.1 on the right: although the hands of user B are outside the field of view of their hand tracking camera, they are tracked by the camera of user A and can be rendered correctly. In this case, the camera of user A is tracking four hands at the same time. This is not possible with the integrated hand tracking of the Meta Quest, where Han et al. present the recognition algorithm with the fact that they expect a maximum output of two hands, making the recognition of more than two hands impossible [37]. The same applies to LeapMotion or integrated hand detection systems of other HMDs (like the HTC Vive Cosmos).

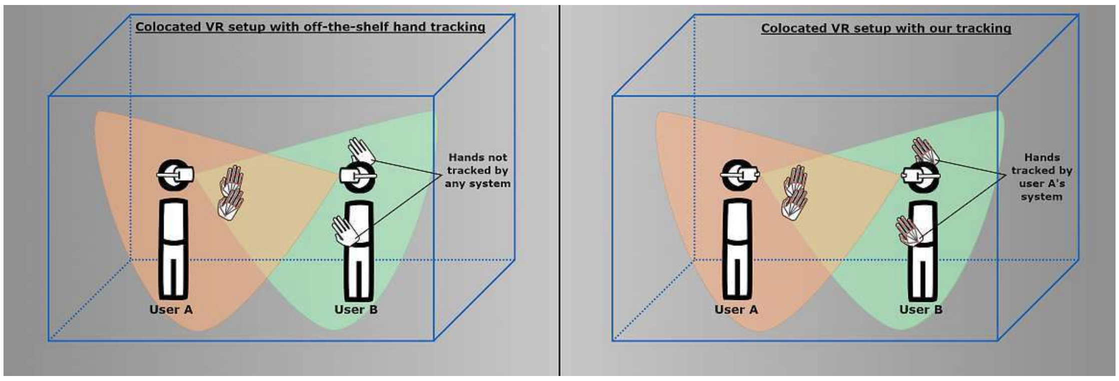

Colocated VR setup with off-the-shelf hand tracking
Hands not tracked by any system
User A
User B
Colocated VR setup with our tracking
Hands tracked by user A's system
User A
User B

Figure 5.1: A colocated multi-user VR setup. User B's hands are outside the range of their own hand tracking, but visible to user A. With off-the-shelf solutions, only two hands can be detected at the same time within a short range (left). Our solution allows us to detect and position the hands of other users in 3D space. This way the hands of user B can still be detected (right).

Since hand tracking methods integrated into off-the-shelf devices are closed systems, it is currently impossible to adjust them to closely align with the interaction requirements of colocated multi-user VR. For this reason, we turn to methods that use RGB input to detect the user's hands. Camera-based methods have certain advantages: they can work with any RGB source, not being bound to any specific hardware, and they work at larger distances, the limits of tracking being set only by the resolution of users' hands in the images. However, most RGB-based solutions offer the capability of detecting the hand pose in 2D image coordinates only, additional calculations being necessary to obtain the full 3D pose.

This chapter presents a hand tracking method that is based on the MediaPipe framework [64][130], a cross-platform solution for object recognition (including hand recognition) in 2D images using machine learning. With this framework it is possible to recognize more than two hands at the same time (see Figure 2.4), which qualifies it for use for our mentioned purposes. To calculate the full 3D hand pose based on finger joint coordinates provided by MediaPipe, we have developed an algorithm that uses an estimation of the user's hand size to obtain its distance from the tracking camera.

We evaluate the performance of our method in comparison with hand tracking methods provided by Meta Quest and LeapMotion, providing an accuracy assessment for each method in static and dynamic conditions in the range from 0.25m to 3m from the tracking camera. With these results we want to determine whether our proposed method provides comparable or even better tracking accuracy in different tracking ranges. With good tracking accuracy at typical tracking area-scale distances and the ability to track more than two hands, our method would present a step towards enabling reliable natural hand interactions in colocated multi-user VR.

5.2 3D World Position Estimation of a Camera Tracked Hand

Our proposed method for enhancing the tracking data of the RGB tracking system consists of three stages:

1. **Hand Detection:** Hands are detected in the monocular RGB image; 3D positions of finger joints are calculated relative to the center of each hand. This stage is carried out by the hand tracking implementation of Zhang et al. in the MediaPipe framework [130].
2. **Hand Size Estimation:** Real-world hand size of the user is estimated according to one of the methods described in section 5.2.2.
3. **Depth Estimation:** Distance of the hand to the tracking camera is calculated according to the method described in section 5.2.3, using the estimation of the real hand size.

This workflow is presented in Figure 5.2, which includes the details of each stage described in the sections below.

Our objective is to develop a workflow for calculating the hand sizes of users and utilizing this information to accurately position the tracked virtual hand (using the MediaPipe framework) within a three-dimensional environment, thereby enabling natural hand interactions. To evaluate the effectiveness of our approach, we will compare it with existing off-the-shelf tracking solutions.

By leveraging hand size estimation for positioning, we anticipate higher accuracy in hand tracking as the precision of hand size estimation improves. We also expect to achieve accurate hand size estimation by inferring the hand size from the user's body height (derived from Pheasant [90]).

Overall, expect a system capable of facilitating 3D hand interactions with a much larger tracking range, surpassing the capabilities of existing off-the-shelf tracking solutions. This advancement will make our solution highly advantageous for use in colocated VR scenarios.

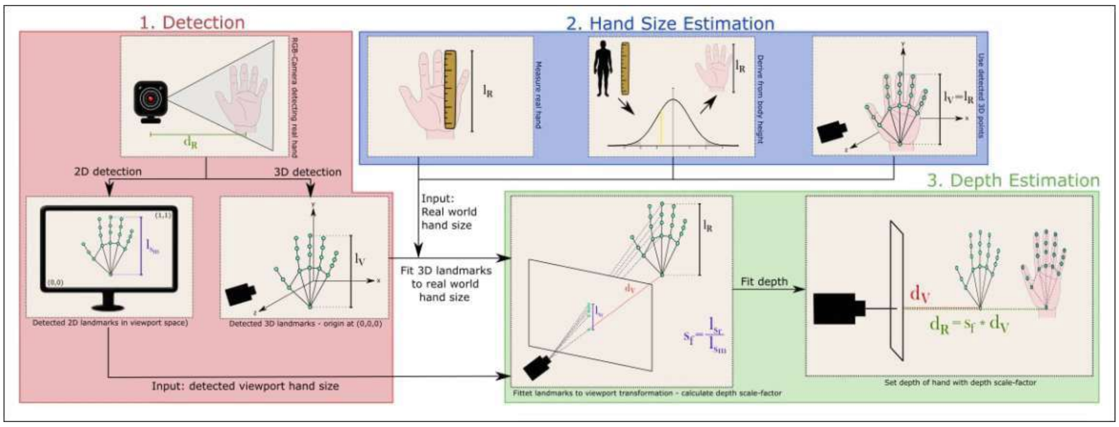

1. Detection
RGB-Camera detecting real hand
d_R
2D detection
3D detection
(1,1)
l_{sm}
(0,0)
Detected 2D landmarks in viewport space)
l_V
Detected 3D landmarks - origin at (0,0,0)
Input: detected viewport hand size
2. Hand Size Estimation
l_R
Measure real hand
l_R
Derive from body height
$l_V = l_R$
Use detected 3D points
Input: Real world hand size
Fit 3D landmarks to real world hand size
3. Depth Estimation
l_R
d_V
l_V
$s_f = \frac{l_{sr}}{l_{sm}}$
Fittet landmarks to viewport transformation - calculate depth scale-factor
Fit depth
d_V
$d_R = s_f * d_V$
Set depth of hand with depth scale-factor

Figure 5.2: Step-by-step diagram for adjusting and positioning a detected hand from MediaPipe. After detection, the virtual hand is adjusted to the real-world hand size and then positioned with the help of the intercept theorem.

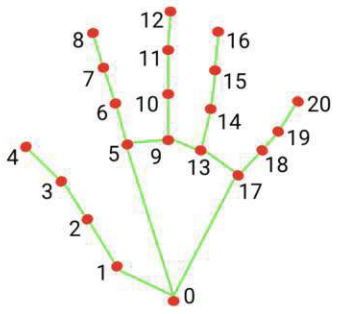
5.2.1 2.5D Joint Detection with MediaPipe

We chose the MediaPipe framework ([64]) as the hand detection step in our workflow (and the hand and finger detection implemented by Zhang et al. [130]) due to its ability to detect more than two hands at the same time. As they report an average precision between 86.22% and 95.7% for palm detection, we can assume a similar detection accuracy in our experiment.

With the help of two TensorFlow machine learning models (palm detector and hand landmark model), MediaPipe tracks the finger joints of the hand with high prediction quality. As a result of the recognition, we get the following information from the framework for each detected hand:

- **Handedness:** A label ('left' or 'right') and an estimation probability for this handedness.
- **World Landmarks:** 21 landmarks (a landmark corresponds to a finger joint) consisting of x, y, and z coordinates with the origin at the hand's approximate geometric center.
- **Normalized Landmarks:** 21 landmarks consisting of x, y, and z coordinates in the normalized viewport space of the camera.

The landmark definitions can be seen in Figure 5.3 (taken from the official MediaPipe hand tracking website [71]). The marked landmarks are later used for hand length calculation in the virtual space. Together with the remaining landmarks in the coordinate frame of the center of the hand and normalized landmarks, they are used in the calculation of the distance of the hand to the tracking camera. This detection step can be seen as the first step in Figure 5.2.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 5.3: Landmark indices of the MediaPipe framework. Marked landmarks are used for hand length calculations.

5.2.2 Estimating Real Size of Users' Hands

We use the real-world length of the user's hand to estimate the distance of the hand to the camera. To obtain the hand's size, three different methods are used, resulting in three variants of our hand tracking method that were evaluated. In Figure 5.2 these methods are visualized in the second step.

1. We use MediaPipe 3D hand landmarks with the origin in the center of the hand to calculate the distance between the position of the wrist and the tip of the middle finger. This distance represents the length of the hand. We refer to this method of hand size calculation as *MediaPipeInternal* in the rest of the chapter.

2. We measure the real length of the user's hand (wrist to the tip of the middle finger) and use the measurement as an input to our program (later referred to as *MediaPipeHand*).

3. The third method requires more calculations but could provide an easier setup experience for the user. Since most people do not know the length of their hand, we use the body height (measured manually for best accuracy) as an input parameter to infer the length of the hand (later referred to as *MediaPipeBody*).

	Men				Women			
Dimension	5th %ile	50th %ile	95th %ile	SD	5th %ile	50th %ile	95th %ile	SD
1. Stature	1625	1740	1855	70	1505	1610	1710	62
⋮								
28. Hand length	175	190	205	10	160	175	190	9
29. Hand breadth	80	85	95	5	70	75	85	4
⋮								

Figure 5.4: Body Size estimations excerpt from Pheasant [90].

Pheasant conducted an examination of different body part sizes and their frequency in the English population [90]. Figure 5.4 is an excerpt from this book and shows different estimations of the size of body parts (in mm) with three percentiles (including the mean) and the standard deviation for a normal distribution. We use these values to create normal distributions and derive body part sizes from another reference body part size.

Zafar et al. evaluated the body-hand relations by calculating the body size based on the size of the hand with an accuracy of 2.9 cm [127]. Since their method also requires the age of the user and we want to keep the input data set as small as possible, we calculate the hand size using the tables from [90].

In this way, the actual body part does not have to be physically measured. In our case, we use the body size of the user to get its percentile in the normal distribution which is then used to recalculate the hand length size for this percentile. For this we use the following equations:

$$\begin{aligned} z &= \frac{x_0 - \mu}{\sigma} \\ p &= \frac{1}{2}\left[1 + erf\left(\frac{z}{\sqrt{2}}\right)\right] \\ x &= \mu + (\sigma * z) \end{aligned} \tag{5.1}$$

For our experiment in section 5.3.2, our user had a body size (height) of 1892mm. For this size we look up $\mu = 1740$mm (given at a percentile of 50%) and $\sigma = 70$ in Figure 5.4. With Equation 5.1 we calculate a percentile of *0.985*.

For the hand size we look up $\mu = 190$ mm and $\sigma = 10$. With the percentile of *0.985*, we can estimate a hand size of *211.71 mm* for the given body size. In comparison, we have determined a measured hand size of *213 mm* for the user.

We calculated the size of the hand based on the body height of all participants in our user test ($n = 10$, see section 5.3.2 & section 5.3.3) and compared the resulting value with the measured size of the hand.

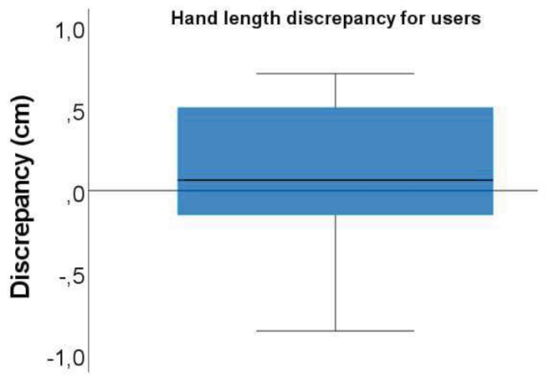


Figure 5.5: Discrepancy between measured hand length and calculated hand length of ten users that participated in the evaluation of section 5.3.2 & section 5.3.3.

The box-plot of the difference between the calculated and measured hand lengths can be seen in Figure 5.5. With a mean difference of 0.0787 cm, it can be seen that the hand length can be calculated accurately from the body size, even though differences of up to 1 cm are possible depending on the user.

5.2.3 Estimating Hand Depth

With the fitted 3D coordinates of the tracked landmarks we now have a correctly scaled hand with the expected hand length, which only has to be adjusted in the distance to the virtual camera. To achieve this we use the intercept theorem ([100]), which describes rules about the ratio of parallel line segments which are intersected by a line. We calculate the hand length of the normalized 2D landmark positions in the camera's image space (l_{s_m}), which are obtained from MediaPipe. We also transform our fitted 3D coordinates to the viewport space and calculate the hand length of these transformed viewport points (l_{s_r}).

With the intercept theorem, we know the following about the ratios:

$$\frac{d_R}{d_V} = \frac{l_{s_r}}{l_{s_m}} \tag{5.2}$$

To get the final depth d_R to the camera we solve the equation to $d_R = \frac{l_{s_r}}{l_{s_m}} * d_V = s_f * d_V$ where d_V in this step is the current distance of the virtual hand to the virtual camera. The step-by-step procedure from detecting the hand to virtual positioning can be seen in Figure 5.2.

5.3 Accuracy Evaluation Experiment

We conducted three experiments to evaluate the effectiveness of our hand tracking method. In **Experiment 1**, we conducted a comprehensive technical assessment to compare the accuracy of our method against two off-the-shelf solutions: LeapMotion and Meta Quest. This evaluation focused on the hand data from a single user. **Experiment 2** aimed to validate the performance of our hand tracking method with hand images from multiple users. We analyzed three variants of our method using data input from nine users. In **Experiment 3**, we conducted a demonstrative test to assess the application of our hand tracking method in an actual colocated scenario, involving a pair of users.

The following sections will present each experiment sequentially, along with their corresponding results. This organization facilitates a clear grouping of each experiment with its respective results, enhancing readability and understanding of the outcomes.

5.3.1 Setup

As hand tracking devices we used a LeapMotion sensor, a Meta Quest 2 HMD with its integrated hand tracking, and a 1080p webcam with a 60° horizontal field-of-view and with MediaPipe as the tracking framework. Figure 5.6 shows a sketch of the experimental setup.

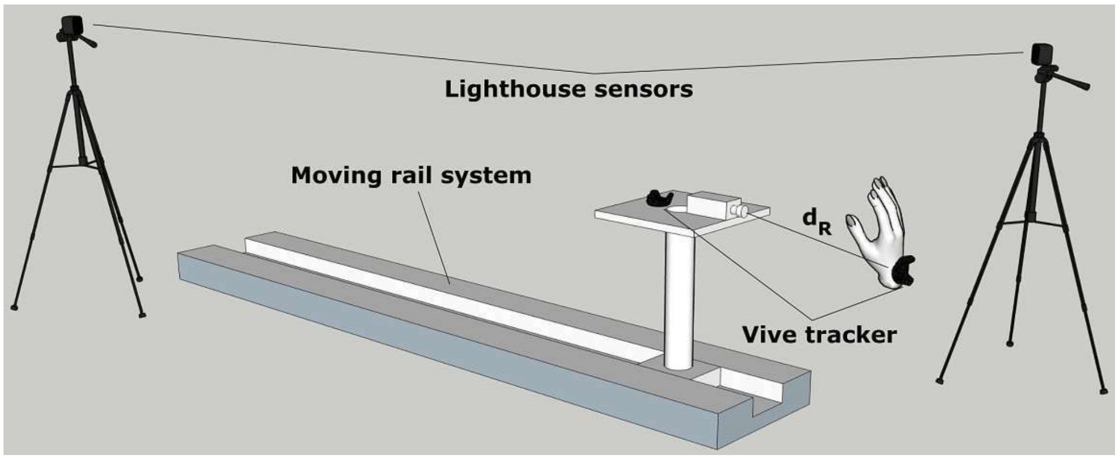


Figure 5.6: Sketch of the experimental setup.

The tracking devices were mounted on a fixture that can move back and forth along a rail. The Vive Lighthouse system was used to calculate real-world distances. The real-world position offset between the Vive tracker and the real-world position of the hand tracking device (as well as the offset between the Vive tracker and the real hand) was measured and taken into account in the virtual world calculations. The rail is 4 m long, which was sufficient for the maximum tracking distance of the evaluated methods. Vive Lighthouse base stations were positioned around the rail system. One Vive tracker was attached to a fixed position on the bracket, and the other to the wrist of the hand to be detected. The offsets to the tracking devices and the center of the hand were measured and added to the respective positions in the evaluation.

The VR application to collect evaluation data was developed with Unity3D (v.2021.2.14). The rendering of tracked user hands was achieved with the help of the 'EasyHand' framework that was presented in chapter 3. For hand tracking, the following versions of the hand tracking API were used: Oculus Integration v.38, Ultraleap Plugin v.5.4.0 and MediaPipeUnityPlugin[1] v.0.8.3 with MediaPipe backend v.0.8.9.

5.3.2 Experiment 1: Comparison to Integrated Tracking Solutions

In this experiment, we access the performance of our hand tracking method based on RGB input by comparing it to the methods integrated into Meta Quest and LeapMotion. All selected solutions can be combined with HMDs and are therefore, in principle, suitable for use in a colocated VR setup. We evaluate three variants of our RGB-based hand tracking method (as explained in section 5.2.2: **MediaPipeInternal**, **MediaPipeHand** and **MediaPipeBody**.

[1]Mediapipe Unity Plugin: `https://github.com/homuler/MediaPipeUnityPlugin` (Accessed: 2023-12-15)

Since our MediaPipe-based method is not fine-tuned for a specific interaction range, in contrast to LeapMotion and Meta Quest, we do not expect it to be more accurate than those methods in the close range (within the arm's length). Nevertheless, we expect to be able to cover a larger tracking range with MediaPipe hand tracking and to achieve higher tracking accuracy at distances beyond arm's length. We expect our method to provide a usable tracking capability that works with a simple RGB camera, is simple to set up and has the ability to detect more than two hands at a time.

We analyze the following metrics:

- Static distance error: Error in the distance of the hand from the tracking device, compared to the ground-truth, while the hand is held still in one position. This metric is calculated at different distances from the tracking device.
- Dynamic distance error: Error in the distance of the hand from the tracking device, compared to the ground-truth, while the hand is moving away relative to the tracking device. We analyze the correlation between the dynamic error and the distance to the tracking device for all methods.
- Tracking lost and tracking acquired distances: Distance at which tracking is lost when the hand is moving away and the distance at which tracking is found when the hand is moving towards the tracking device.

The ground-truth distance d_r of the hand to the tracking device is measured with an externally mounted Lighthouse 2.0 tracking system, the tracking accuracy of which has been shown to be in the millimeter range with high replicability of position measurements ([4][7]). HTC Vive trackers were attached to the wrist of the hand and the tracking device, allowing the ground-truth distance d_r in the frame t to be calculated as the distance between two trackers, adjusted by the offset between the center of the tracker and the center of the hand and between the center of the second tracker and the center of the tracking device.

The virtual distance d_v from the virtual hand to the virtual camera is calculated from the hand tracking data. The absolute euclidean distance error $\delta(t)$ for a frame t can thus be calculated with

$$\delta(t) = |d_v - d_r| \tag{5.3}$$

Static Error Evaluation

For the static error measurement, the tracking device and the tracked hand of a user were positioned at fixed distances from each other with $d \in 25, 50, 75, 100, 150, 200, 250$ cm. Smaller intervals of 25 cm were chosen in the range where LeapMotion and Meta Quest could consistently track the hand. For $d >= 100$, the sampling interval was increased to 50 cm since only MediaPipe was able to consistently track the hand for these distances. A maximum length of 250 cm was chosen, as LeapMotion and Meta Quest had a much shorter possible tracking distance in initial tests and we felt that this distance was sufficient to evaluate the MediaPipe system and thus keep the evaluation time tolerably short.

At every sampling position, we collected hand tracking data over N=400 consecutive frames. This was done 25 times per position for each tracking method (Quest, LeapMotion, and three variants of the MediaPipe method). Each sample containing tracking data of 400 consecutive frames was collapsed to its median value, to account for possible small movements of the user's hand. This way, our resulting evaluation sample for each tracking method consists of 25 median static error values per one sampling position. Collections were only carried out if there was consistent tracking during the 400 frames. For Meta Quest and LeapMotion this worked in the range [25 cm;75 cm] and for all variations of the MediaPipe method in the range [25 cm;250 cm].

Results A few outliers were observed, which likely resulted from the temporary tracking loss of the deployed ground truth trackers in the environment. As these outliers were not attributed to the tracking of the hand itself, they were excluded from the analysis to avoid any false influence on the results. Following the removal of outliers, the Shapiro-Wilk normality test was performed on all median error samples [102]. Because not all median error distributions were normal and because sampling position ranges were different for the evaluated methods, we compare the median static error of the evaluated methods separately for each sampling position, using the non-parametric Independent-Samples Median test. We also analyze the impact of the distance to the tracking device on the median static error for each method in the non-parametric Friedman's 2-way ANOVA test [25][26][27]. The corresponding box-plots are presented in Figure 5.7.

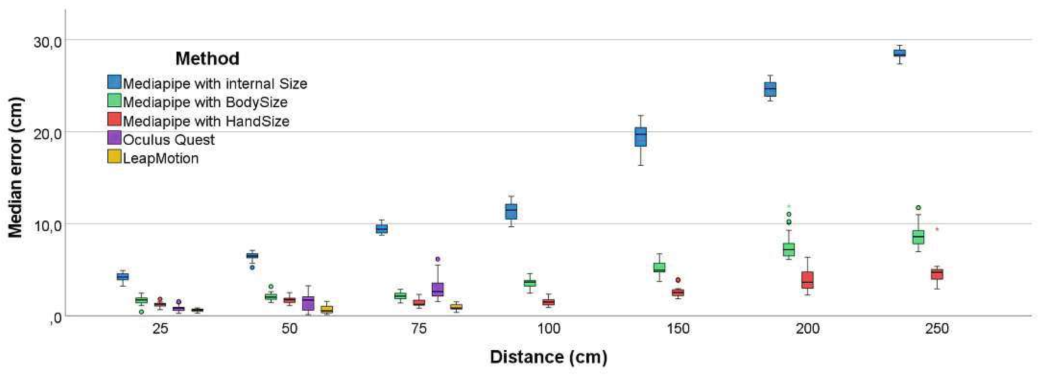


Figure 5.7: Box-plots for static data collection for each method and distance.

As expected, the tracking error for close distances [25 cm;50 cm] is the lowest for Meta Quest and LeapMotion. Interestingly, MediaPipeHand delivered a lower error of 1.53 cm than Meta Quest with a distance of 75 cm to the camera ($p < 0.001$), which is still within the user's arm's reach. Except for LeapMotion, the other tracking methods show an increasing median error as distances are increased to the camera. This is in line with our results from the dynamic evaluation.

For distances greater than 75 cm, only the values of the MediaPipe methods can be compared. However, a significant difference in mean error can be found between all three methods for all distances ($\mathbf{p < 0.001}$). The error for MediapipeInternal is significantly larger at all further sampling positions than for MediaPipeBody and MediaPipeHand. For large distances, the error of MediaPipeInternal increases to values significantly above 10 cm, which can be too inaccurate for precise interactions. The results show that this error can be significantly reduced by using the real hand size ($\mathbf{p < 0.001}$). With a maximum error of 4.47 cm at a distance of 250 cm from the camera, reasonably precise interactions in virtual space at greater distances are also possible with this method. The derivation of the size of the hand by the size of the body also shows a significantly lower error. However, as expected, the accuracy is not quite as good as when the actual hand size is given. With a maximum error of 8.57 cm at a distance of 250 cm from the camera, the error is significantly larger, but still improves the initial recognition of MediaPipe by more than three times (with an initial error of 28.48 cm). A summary of the resulting mean and median values can be found in Table 5.1.

	25cm					50cm					75cm					100cm				
	MPI	MPB	MPH	LM	Q	MPI	MPB	MPH	LM	Q	MPI	MPB	MPH	LM	Q	MPI	MPB	MPH	LM	Q
Mean (cm)	4.14	1.71	1.81	0.58	0.73	6.53	2.01	1.71	0.67	1.49	9.45	2.14	1.42	0.93	2.94	11.41	3.63	1.52	-	-
Median (cm)	4.2	1.71	1.25	0.6	0.76	6.53	1.99	1.73	0.53	1.71	9.40	2.13	1.24	0.83	2.58	11.49	3.66	1.5	-	-

	150cm					200cm					250cm				
	MPI	MPB	MPH	LM	Q	MPI	MPB	MPH	LM	Q	MPI	MPB	MPH	LM	Q
Mean (cm)	19.27	5.25	2.41	-	-	24.62	6.95	3.85	-	-	28.49	8.57	4.47	-	-
Median (cm)	19.72	4.94	2.46	-	-	27.7	6.89	3.64	-	-	28.37	8.47	4.71	-	-

Table 5.1: The resulting mean and median values for all distances in the static error evaluation. Error values are in centimeters.

These results show that the accuracy of MediaPipe tracking in 3D space can be significantly improved by inputting the user's real hand size and allowing 3D interactions for large distances, which is not possible for the LeapMotion sensor or the Meta Quest hand tracking.

Dynamic Error Evaluation

For the dynamic evaluation, the user held his hand at a fixed position (resting his elbow on a fixture to guarantee consistent height of the hand) while the tracking device moved away from it on a rail system (starting at the distance of 25 cm), at a constant slow speed up to a distance at the edge of the device tracking range. The user had extended all fingers, as this is an easy to hold and repeatable gesture and is usually well recognized by hand recognition systems. While a user walking away from the tracking device would have presented a more ecologically valid hand tracking situation, the rail system was used to ensure repetitiveness and uniformity of data collection for this technical evaluation.

The movement range was: for the LeapMotion sensor $r \rightarrow [0.25; 0.75]$, for Meta Quest $r \rightarrow [0.25; 1.75]$ and for MediaPipe $r \rightarrow [0.25; 2.75]$. Interestingly, the Meta Quest achieved a higher tracking range here than in the previous test, which may be due to the fact that the system can recognize an already recognized hand for longer than it takes to recognize a completely new hand. However, the real reason for this remained hidden from us. This procedure was repeated 10 times for each tracking method. Dynamic error data resulting from these recordings were averaged and analyzed according to a procedure described in detail in the following section.

Results For each frame, we get the real-world distance d_r and the virtual world distance d_v of the hand to the camera. Tracking error was calculated with Equation 5.3 and paired with the real-world distance d_r.
Examples of dynamic error data samples for Meta Quest and MediaPipeHand prepared in this way are illustrated in the scatter plot in Figure 5.8.

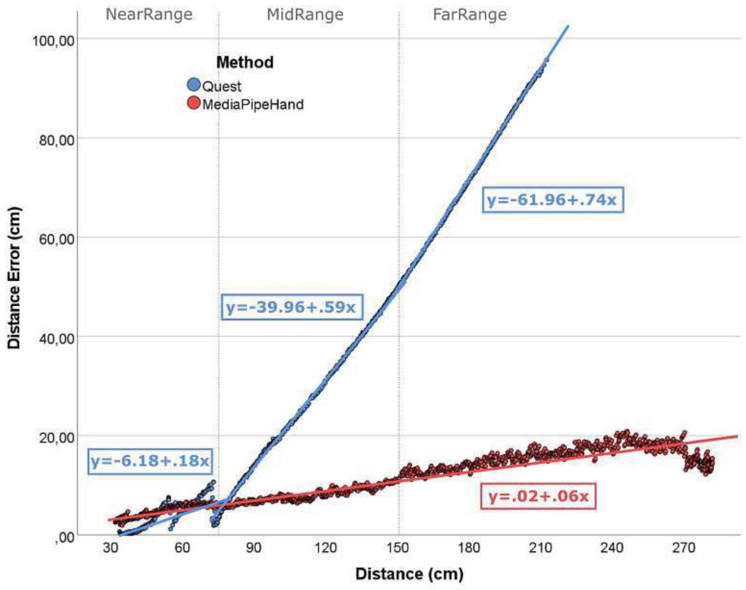


Figure 5.8: Scatter plots of two example dynamic error distributions for Meta Quest and MediaPipeHand. The x-axis refers to real-world distance of the hand to the tracking device, the y-axis refers to the error difference between real-world and virtual-world distance. Three Regression lines for Meta Quest data are illustrated separately for **NearRange**, **MidRange** and **FarRange** due to the rising gradients in each range. One regression line for MediaPipeHand over all distances is illustrated. The linear equations for the regression lines are in $\frac{cm}{cm}$. Quest median error shows a higher rising error for large distances.

The discretized dynamic error distributions are used to perform linear regression, with the gradient of the fitted regression line determining the rate of the error increase with distance from the tracking device. Since the tracking ranges of the evaluated tracking methods are different, we perform the linear regression separately in three distance ranges: **NearRange** [< 75 cm], **MidRange** [75 cm;150 cm] and **FarRange** [>150 cm]. The NearRange distance was chosen based on the results from the previous experiment, as both LeapMotion and Meta Quest delivered reliable results here. We are also within the usual range of one arm's length. MidRange was also selected as Meta Quest provided additional results here in the current experiment. FarRange was then selected for all results above MidRange, which was only achieved by the MediaPipe methods in the experiment (the Quest did reach the range a little, but already in error values that were not in relation to the other methods). This procedure results in 10-entry distributions of regression coefficients for every tracking method, in distance ranges covered by the method. We can now compare all five tracking methods in **NearRange**, Meta Quest and three MediaPipe variants in **MidRange**, and three MediaPipe methods in **FarRange** (the analysis for the Quest was carried out in this range as well for the sake of completeness). We use one-way ANOVA to compare mean regression coefficient values between the methods within each distance range (data is normally distributed in the Shapiro-Wilk test).

The resulting ANOVA plots are shown in Figure 5.9. The scaling of the gradients is $\frac{error}{distance}$ given in $\frac{cm}{cm}$, which shows the increase in the tracking error in cm per each cm of distance from the tracking device.

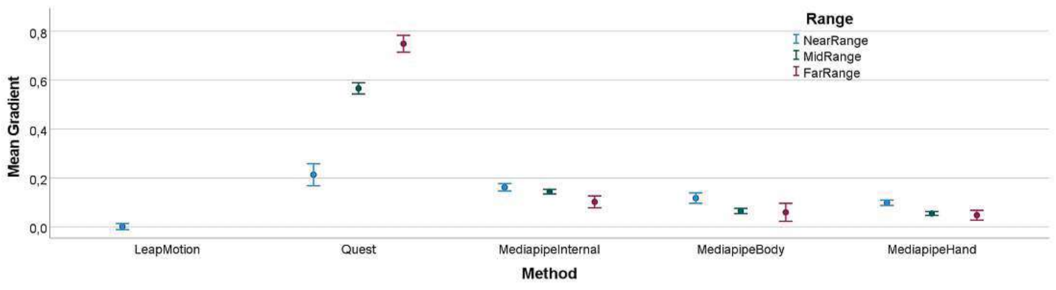


Figure 5.9: Mean gradient values for each method and distance range. Gradients are in $\frac{cm}{cm}$.

In **NearRange**, LeapMotion shows the most consistent tracking with a mean gradient of 0.0007. This is a significantly ($\mathbf{p < 0.001}$) smaller error increase rate than for the Meta Quest and the MediaPipe methods. Meta Quest has the steepest error increase rate with distance with a mean gradient of 0.213 and shows significant differences to MediaPipeBody ($\mathbf{p = 0.006}$) and MediaPipeHand ($\mathbf{p = 0.002}$). The mean gradient of 0.098 for MediaPipeHand is the lowest for the three MediaPipe methods, with a mean gradient of 0.117 for MediaPipeBody and 0.162 for MediaPipeInternal. The statistical analysis, along with a substantial effect size of $\eta^2 = 0.827$ (as measured by η-squared), reveals significant differences among all MediaPipe methods, except for MediaPipeHand and MediaPipeBody. Further information on means and pairwise comparisons can be found in Table 5.2.

MidRange shows significant differences ($\mathbf{p < 0.001}$) and an effect size of $\eta^2 = 0.992$

for all pairwise comparisons of mean gradients, except (as in **NearRange**) between MediaPipeHand and MediaPipeBody (p = 0.333). With a gradient of 0.567, the Quest has the largest gradient here, which more than doubled compared to **NearRange**. Since the gradient of MediaPipeHand with 0.054 and MediaPipeBody with 0.064 has decreased somewhat compared to the **NearRange**, the tracking error in these areas does not increase as much. In comparison, the gradient of MediaPipeInternal with 0.144 is similarly high as in **NearRange**. This shows that the external input of hand size (whether measured or by body size) improves the tracking error. A significant difference between hand size by measurement and by body size cannot be found in **MidRange**.

In **FarRange**, the Meta Quest again shows a higher gradient compared to the closer ranges (with a mean gradient of 0.749). This is again significantly higher than in the MediaPipe methods (**$p < 0.001$**). This shows a strongly increasing error for Meta Quest and large distances and that the hand tracking of the Quest does not seem to be aligned for these distances. The MediaPipe methods again show similar mean gradients as in **MidRange**. With a mean gradient of 0.102, MediaPipeInternal shows the largest change in tracking error among the 3 methods, with a significant difference from MediaPipeHand (**$p = 0.006$**; a significant difference from MediaPipeBody could not be shown at p = 0.170). With a mean gradient of 0.059 for MediaPipeBody and 0.047 for MediaPipeHand, these two gradients are close to each other. As there are again no significant differences for **FarRange** between these two (p = 0.915), it can generally be stated that the increasing error can be significantly improved by entering the size of the hand compared to the size of the internal hand, but no significant differences could be determined between the calculation of the hand size by measurement and the derivation by body size. The one-way ANOVA analysis yielded an effect size of $\eta^2 = 0.98$, as measured by η-squared. This substantial effect size indicates a strong influence of the methods on the observed differences in the gradients. The findings highlight the significant impact that the choice of method has on the measured outcomes.

Figure 5.8 shows the distributed points with its linear regression lines, where one can see how fast the error rises for Meta Quest in comparison to the MediaPipe method with inputting the measured real hand size.

	NearRange					MidRange					FarRange				
	MPI	MPB	MPH	LM	Q	MPI	MPB	MPH	LM	Q	MPI	MPB	MPH	LM	Q
Mean (cm)	0.162	0.117	0.0982	0.001	0.213	0.147	0.0644	0.054	-	0.567	0.102	0.059	0.047	-	0.749

Table 5.2: The resulting mean value gradients for different ranges in the dynamic error evaluation. Mean error gradient values are in $\frac{cm}{cm}$.

Lost and Acquired Tracking Distances

For each tracking method, we recorded the distance at which the tracking device loses the hand while being moved away as well as when the tracking device firstly acquires the hand while being moved towards the device.

To determine the lost tracking distance, the user positioned his hand at a distance where it was reliably tracked. After the hand was tracked for at least one second, the device was moved away from the hand and distance at the moment in which the device lost the hand was recorded. To avoid recordings for moments in which the tracking was only briefly disrupted, the hand had to be lost for at least one second.

The measurement of the distance at which the tracking device acquires the tracking of the hand follows a similar procedure. The device was set at a distance where the hand is not detected (LeapMotion: 1m; Quest: 1m; MediaPipe: 3.75m). After the program ensured that the hand was not tracked for at least one second, the device was moved toward the hand until the hand was tracked. Again, the hand has to be tracked for at least one second to record the distance. Both procedures were repeated 25 times per method.

Results A total of 25 data points were collected for each tracking method and both evaluations.

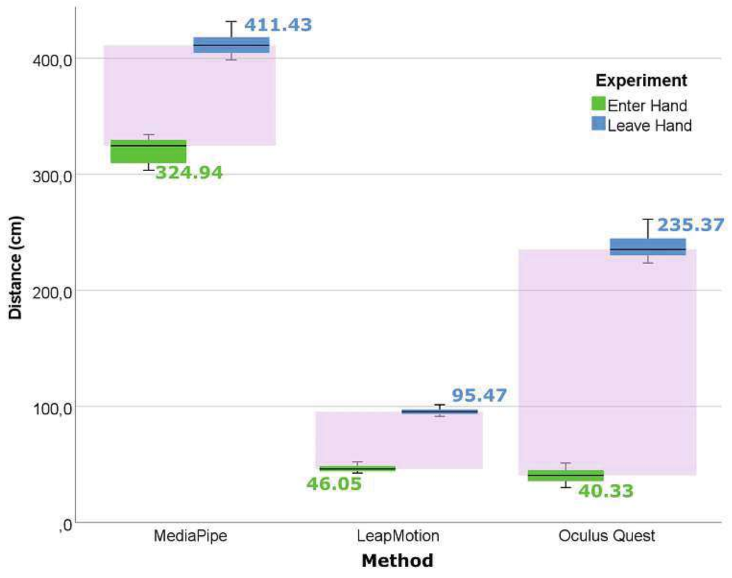


Figure 5.10: Box-plots medians for lost and acquired tracking distance for tracking methods. Labels are median values.

The medians for **lost tracking distances** proved to be normally distributed in the Shapiro-Wilk test ([102]) after three outliers (caused by interferences in the ground truth tracking) had been removed (p = 0.229 for the MediaPipe tracking, p = 0.557 for the LeapMotion, p = 0.293 for the Quest tracking). The resulting box-plots can be seen in Figure 5.10.

For the main analysis, Welch's test ([123]) was employed, yielding a significance level of $\mathbf{p < 0.001}$ and an effect size of $\eta^2 = 0.827$. Subsequently, the post-hoc analysis was conducted using the Games-Howell test [30]. The findings from the post-hoc analysis demonstrated that the mean lost tracking distance for the MediaPipe tracking method (mean distance of 412.48 cm) was significantly greater than that of all other methods ($\mathbf{p < 0.001}$). The mean lost tracking distance of the Meta Quest was significantly larger than with LeapMotion ($\mathbf{p < 0.001}$), but also significantly lower than with MediaPipe tracking ($\mathbf{p < 0.001}$). The lowest mean distance where the tracking is lost is the lowest for LeapMotion with 95.6 cm. With 237.78 cm the Meta Quest has quite a large tracking range, but also has severe tracking errors at longer distances, as can be seen in the previous sections. The detailed results can be found in Table 5.3.

Distributions of recorded distances at which **tracking was acquired** deviated from normal (in the Shapiro-Wilk normality test) for two out of three tracking methods; therefore, median tracking acquired distance values were compared in the Independent-Samples Median test. The corresponding box-plots are illustrated in Figure 5.10.

The pairwise comparisons of the Independent-Samples Median test demonstrate significant differences between the methods, with a notable effect size (as measured by η-squared) of $\eta^2 = 0.827$. These comparisons reveal that the median distance of 324.94 cm for MediaPipe is significantly greater than that of LeapMotion, which measures 46.05 cm ($\mathbf{p < 0.001}$), as well as Meta Quest, with a median distance of 40.33 cm ($\mathbf{p < 0.001}$). The difference in acquired tracking distance between Meta Quest and LeapMotion is also statistically significant ($p = 0.008$), although the corresponding median values are much closer (with a difference of 5.72 cm). The fact that MediaPipe consistently acquires hand tracking in a range of approximately 3 meters shows that this method could be used for larger areas while LeapMotion and Meta Quest are limited to near-range distances within arm's length.

	EnterHand			LeaveHand		
	MediaPipe	LeapMotion	Quest	MediaPipe	LeapMotion	Quest
Mean (cm)	321.51	46.6	40.27	412.48	95.6	237.78
Median (cm)	324.94	46.05	40.33	411.43	95.47	235.37

Table 5.3: The resulting mean and median distances for the different tracking methods when tracking is acquired and lost. Distance values are in centimeters.

5.3.3 Experiment 2: Accuracy of MediaPipe-Based Hand Tracking for Multiple Users

The primary objective of this second experiment was to validate the effectiveness of our MediaPipe-based method for various users and to assess the influence of hand size estimation on the accuracy of the 3D positioning error in greater depth. Although it did not involve an extensive user study with joint-error measurements, it served to confirm the usability of our method and highlight other impacts on positioning errors. The user test is intended to validate the applicability of our proposed methods for calculating hand length and positioning the hand in 3D space using various input data (such as body height and hand size).

In this experiment, we repeated the procedure for static and dynamic tracking error measurement from **Experiment 1** for each user separately, this time focusing only on MediaPipeInternal, MediaPipeBody, and MediaPipeHand methods. For the static error measurement, four hand tracking data recordings (400 frames each) per distance per user were made at distances $d \in 25, 50, 75, 100, 150, 200, 250$ cm between the tracking device and the user's hand, resulting in four median error values. One recording of dynamic tracking error data per user (for each method) was conducted. The calculations of the distance error in the static and dynamic conditioned were the same as in the previous experiment.

The hardware and software setup used for data recording was the same as in **Experiment 1**. In total, hand tracking data was collected from 9 additional users, 3 female and 6 male, ranging in age from 20 to 56 years. They consisted of students from various fields of study, as well as acquaintances. They took part in the experiment voluntarily and did not receive any compensation. Including measurements of body and hand size and introduction, the procedure took about 45 minutes per user.

Results - Static Error Figure 5.11 presents box plots illustrating the average static error of MediaPipeInternal, MediaPipeBody, and MediaPipeHand at different distances. To maintain a consistent analysis metric and account for the relatively low variance observed in the four median error values obtained for each user at each distance, we consolidated these errors by calculating their median value and employed this value for the analysis. As evident from the results, outliers are observed, which may have been caused by temporary tracking losses. Despite their presence, these outliers were retained in the analysis because it could not be guaranteed that their removal would be advantageous for maintaining the integrity of the test, as previously mentioned.

With the data deviating from the normal distribution in many cases, we again used the Independent-Samples Median Test to find whether the error depends on the hand tracking method at each distance. The method was statistically significant for the error at the distance of 200 cm ($\mathbf{p = 0.016}$; follow-up pairwise comparisons did not find any statistical significance), with the result at the distance of 250 cm being marginally below statistical significance. From the pattern seen in the box-plots with MediaPipeInternal producing visibly higher errors at larger distances, we believe that more detailed results could be achieved with a larger sample size of users or recorded data. Nevertheless, the available data supports the results of the main analysis.

Analyzing the relationship of error to distance from the tracking camera for each method, we found that the error increases with the distance for MediaPipeInternal ($\mathbf{p = 0.002}$ in Independent-Samples Median Test) and MediaPipeBody ($\mathbf{p = 0.029}$). For MediaPipeHand, no statistically significant dependency of an error on the distance could be found.

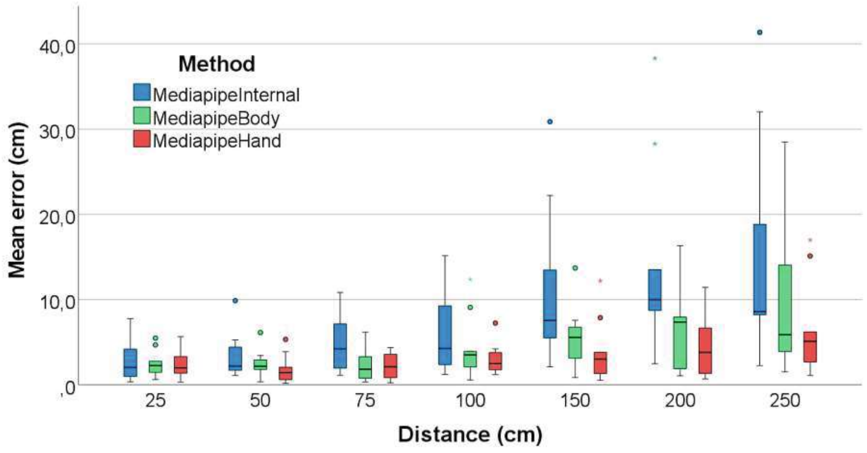


Figure 5.11: Box-plots of the mean error at every measured distance in the test with multiple users.

The results of the previous experiment are confirmed by these results that the tracking error can be improved by entering the real hand size and thus interactions in 3D space can also be made possible over larger distances.

Results - Dynamic Error We used the same approach for calculating the gradients of error change in the near, middle and far range on the evaluation data of multiple users. Figure 5.12 shows the gradients for each method in **NearRange**, **MidRange** and **FarRange**.

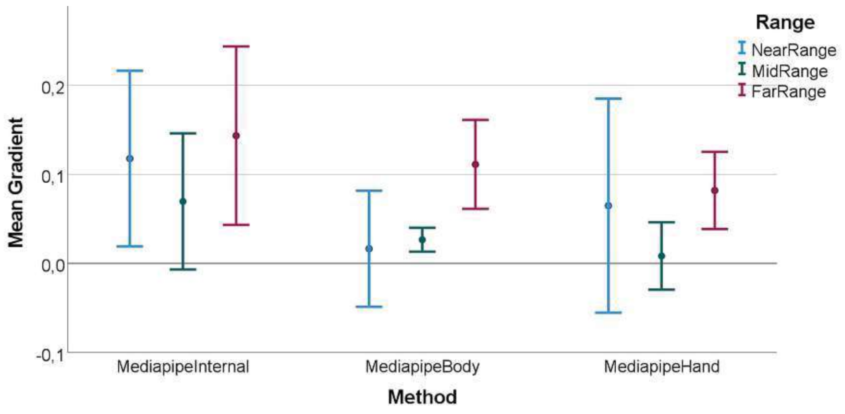


Figure 5.12: Mean gradient values for each method and distance range in **Experiment 2**. Gradients are in $\frac{cm}{cm}$.

We used Mixed ANOVA with the range as a repeated-measures factor (with three levels) and the method as a between-subject factor (also with three levels). In this analysis, only the range was found to be statistically significant (F = 4.683, **p = 0.014**), with within-subject repeated contrasts showing the increase of gradient from **MidRange** (mean = 0.035) to **FarRange** (mean = 0.112), **$p = 0.002$**. The result shows us that the increase in error also increases with greater distance for all MediaPipe methods. Figure 5.12 shows mean gradients for each tested range and method.

5.4 Demonstration in a Colocated Setup

The goal of this last experiment was to test the usability of our hand tracking method in a real-world colocated VR scenario. To do this, we developed a simple VR application in which two users can see each other's avatars consisting of the virtual HMD and hands steered by head and hand tracking input. The aim was not to create a detailed qualitative user analysis, but rather a proof-of-concept demonstration.

In our setup, each user has an HTC Vive tracker attached to their hand, which provides position and rotation data of the real hand positions in the space of the Lighthouse 2.0 Tracking and serves as ground-truth. Both users have a VR headset on (we used Meta Quest with its own hand tracking feature turned off). User 1 has an RGB camera attached to the HMD (in this scenario a ZED mini camera that only provided the RGB image of one lens to use with MediaPipe). With this camera, all hands were detected and visualized in the shared virtual environment. Since only User 1 had a running hand detection system, it was ensured that only one system tracked all hands and placed them in the virtual space. The users stood facing each other at a distance of about 1.5 meters and held their hands in the tracking area of the RGB camera. As a result, the hands of User 1 were in NearRange and the hands of User 2 were in MidRange during the recording. The camera simultaneously detected and tracked the hands of User 1 and User 2. In the virtual environment, the user's hands were positioned with the hand tracking input.

5.4.1 Results

Figure 5.13 shows a snapshot of the scenario experienced by the users in the experiment. The RGB image of the tracking camera is overlayed with the virtual scene seen in VR to give a better illustration of hand tracking. The plane that can be seen in Figure 5.13 is the floor plane in the virtual environment. Since only User 1 had a hand tracking device attached to his HMD, the view of all virtual hands, his own and those of User 2, is enabled by his hand tracking camera. The same virtual hands were displayed to User 2 and animated by hand tracking data.

Hand tracking data was recorded over a period of about 30 seconds, resulting in snapshots of 1242 consecutive frames. The detected hands were assigned in pairs to the corresponding Vive trackers, and the deviation from the position of the ground truth tracker was calculated as a tracking error. The position errors for each user are plotted in Figure 5.14. Peaks and missing values in the plot are moments where corresponding hands were not tracked for a moment.

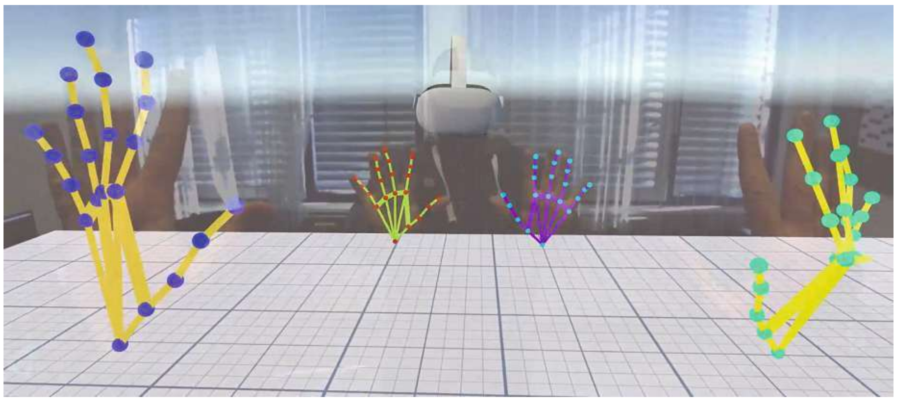


Figure 5.13: The point of view of User 1 with the tracking device attached tracking also the hands of User 2. Both users are colocated, the real-world view of the camera can be seen slightly overlayed.

Since MediaPipe provides a new set of position data every frame, which is not based on previous results, the actual hand movement in virtual space can look very jumpy. Frame-related short interruptions cause virtual hands to jump around a lot in space. To counteract this, we have applied an exponential low-pass smoothing filter that adjusts hand positions based on previous positioning. Short dropouts are thus counteracted and general movements appear much smoother. The implementation is taken from the DigitalRune library and has a time constant of 0.05 ms as a delay [2]. Due to this smoothing filter, inaccurate positioning can occur right before losing or after acquiring tracking.

The calculated mean error for the hands of User 1 was 5.1 cm with a standard error of 0.154. The 95% confidence interval provides a lower limit of 4.8 and an upper limit of 5.41 cm. For User 2, we obtain a mean of 8.21 cm with a standard error of 0.17, a lower limit of 7.88, and an upper limit of 8.55 cm for the 95% confidence interval.

The higher position error values for User 2 were expected, as User 2 was further away from the camera. The values correspond to the expectations of a more realistic scenario based on the results of the dynamic and static tracking error evaluations.

Although we calculated the error of the tracked hands in this scenario, it is important to note that a single user pair is insufficient to provide a comprehensive user study for real-world colocated applications. In addition, the use of direct manual manipulation tasks would be an interesting extension in such a study in order to additionally validate the usability. Nevertheless, we aim to demonstrate that our method is not limited to controlled scenarios with fixed machinery but can also be applied in real colocated setups involving two users. To further evaluate the presented method, it would be valuable to conduct a qualitative user study that measures interaction precision, usability, and user

[2]DigitalRune: `https://digitalrune.github.io/DigitalRune-Documentation/html/81cd4f27-5ce5-4439-9a6c-121f2942f175.htm(Accessed:2024-02-19)`

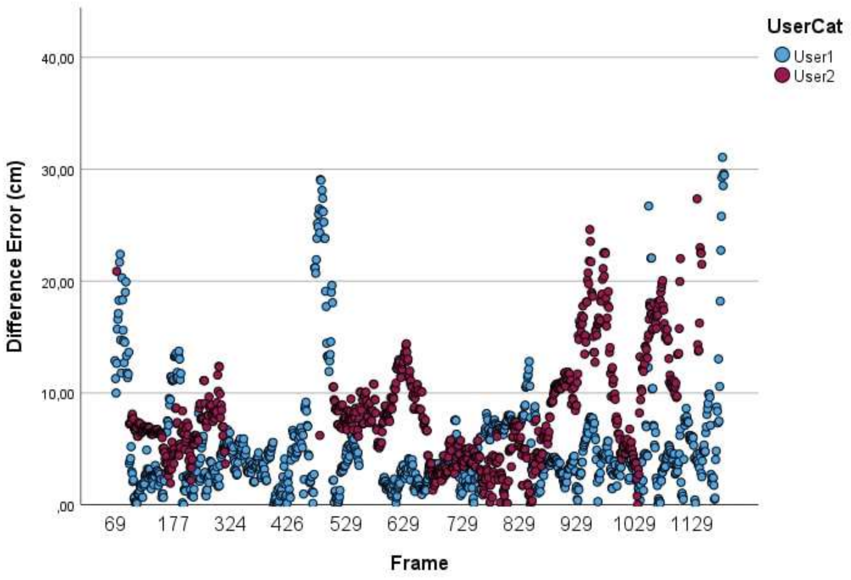


Figure 5.14: Error of hand pairs in centimeters during the preliminary user test. The hands of User 1 were in NearRange, while the hands of user 2 were in MidRange. Spikes can occur due to tracking losses and wrongly positioning due to a smoothing filter.

experience. Such a study would provide an interesting avenue for future research and a more comprehensive evaluation of our approach.

5.5 Discussion

The results of **Experiment 1** show that Meta Quest and LeapMotion are more accurate at arm's length range than RGB input-based tracking with MediaPipe, which was to be expected. The tracking errors for Meta Quest and LeapMotion are in line with previous research results ([1][99]). However, if the user's hand length is used to improve the MediaPipe-based method (either being estimated based on the height or entered from a real measurement), RGB tracking delivers comparable accuracy with a distance error in the range of 12.4mm to 21.3mm (see the lowest and largest error in Figure 5.7 for MediaPipeHand and MediaPipeBody in distances between 25 cm and 75 cm). This shows that in this distance range the improved RGB hand recognition offers interactions with similar precision as the off-the-shelf solutions.

At the best value in the close range of 25 cm, the error of the RGB method with a real measured hand is 3.36 times lower than without external input. In the outer range at a distance of 250 centimeters even by a factor of 6. Due to the improvement through external hand size input, this even allows interactions in ranges of 2.5 meters, which is not possible with Meta Quest or LeapMotion. Assuming a colocated multi-use scenario, the participating users move at different distances from each other. If they carry out joint interactions or tasks at the same location, they move in close proximity, which usually exceeds their own hand length. Detection and interaction ranges of 2.5 meters also qualifies this method for use in such scenarios, although these would need to be investigated in more detail together.

The results of the acquired and lost tracking distance show that Meta Quest and LeapMotion are in a similar range at which distance a hand is recognized for the first time. This makes sense in the sense that both systems were designed for interactions in the arm's length. In comparison, MediaPipe has an almost eight times higher distance at which the hand is recognized for the first time, which also allows for a significantly higher range and more possibilities, for example, to recognize the hands of other users that are further away from the own user.

After a hand has been detected once, all methods have a greater distance where the tracking of the hand is lost again. It is striking that LeapMotion only allows hand detection in the areas where tracking quality is guaranteed to be within a certain tracking confidence range (which LeapMotion calculates), whereas the Meta Quest goes beyond this range and continues tracking the detected hand up to a distance of 235 centimeters. These differences between acquired and lost tracking are shown by the purple 'overtracking ranges' in Figure 5.10. This lack of limitation of the Meta Quest for distances beyond arm's length also leads to very strongly increasing tracking errors in this case. This makes the Quest unusable due to high tracking errors in **Mid-** and **FarRange** detection, even though hand tracking would be possible for the system in this range.

This is also reflected in the results of dynamic tracking. Within the tracking range, LeapMotion has significantly fewer discrepancies in tracking error than, for example, the Meta Quest. The latter shows significantly increasing tracking errors, especially in tracking outside the **NearRange**.

MediaPipe is much more consistent here, even though the tracking error still increases as the distance increases. However, this slope is nowhere near as steep as in Meta Quest. Even between the different presented depth estimations for MediaPipe, the increasing error is significantly lower in the variants where the user's hand size was added externally. Thus, the error is less at larger distances. This is also reflected in the results of the static evaluation. This is a further indication that RGB methods (such as MediaPipe) can be improved and are thus suitable for enabling hand tracking and hand interactions even at greater distances.

During the demonstration, we observed that the computation time and the occurrences of temporal tracking losses increased as the number of simultaneously detected hands increased, although we did not collect accurate real-time data. This behavior aligns with expectations for an image-tracking system designed for multi-object recognition. While the presence of four hands within the tracking frame did not cause significant issues, the limitations of the current state of the MediaPipe system became more apparent as the number of hands increased. However, it is important to note that these observations did not affect our calculations, and we anticipate that future versions of the framework will address these limitations and enhance stability.

As has also been shown, the way the real hand size is determined has a significant impact on tracking accuracy. The more accurately the hand size is determined, the smaller the error. Unfortunately, the 3D coordinates of the finger joints provided by MediaPipe did not accurately match the real hand size and the size of the hand had to be input externally to effectively improve depth calculation.

The results of **Experiment 2** largely reflect the pattern of the main analysis, although significance was not shown in all comparisons. The error of the 3D hand positioning increases with the distance to the tracking camera. However, the error can be reduced (especially for high distances) by inputting the users' hand size. We believe that a larger data set and extended user testing (which unfortunately was not possible at this time) would increase the significance and further increase the level of detail in the results.

Furthermore, it can be seen that when estimating hand size from body size for 10 users (from **Experiment 1** and **Experiment 2**), the deviation of the calculated hand size to the actual hand length was less than 1 cm for all users. The algorithm can be used as an alternative when the user's body size is known rather than their hand length. For future applications, it would be interesting to find a method to perform this measurement more universally and without external input. One possibility would be a calibration step at the beginning of the application, where the hand is measured at a certain distance, or where the body size is determined based on the height of the VR headset and then the hand size is derived. For applications where full-body avatars are used, for example, these could also be used to improve the tracking error that still exists. If they are scaled to the user's size, their maximum arm length can be used as an improvement metric to position the hand more accurately to the avatar. An approach such as 'Virtual Caliper' by Pujades et al. would be interesting and helpful in this respect, that use a VR controller to take measurements in order to adapt a 3D avatar to the user's body measurements [94].

The mean hand position errors calculated in **Experiment 3** are in line with the results of the controlled static and dynamic tests. With continuously detected hands, consistent positioning can take place. In the test, we noticed that the quality of positioning is also dependent on the quality of the underlying tracking (MediaPipe in this case). A more consistent and better recognition in the future also improves the real error of our algorithm. Coupled with a high-resolution camera, consistent multi-user hand tracking in colocated rooms can be realized.

5.6 Conclusion

The study that was presented in this chapter evaluated the tracking error and tracking range of three different hand tracking technologies, one of which works via RGB cameras and is not tied to a specific manufacturer's hardware. In addition, a method was developed that significantly improves the tracking result of RGB hand tracking by taking hand size into account.

The evaluation shows that although the hand tracking of Meta Quest and LeapMotion is more accurate in the arm's length range, they are not designed for hand detection outside this range. Therefore, we achieve comparatively higher precision at larger distances with the RGB method. This can be further increased if additional information about the user's hand length is used in the calculation of the 3D positions. Direct measurement of the hand length is more precise, but deriving the hand length from the user's height also produces comparable tracking errors. For more general use, the body height input is probably more intuitive, as many people know their body height rather than their hand length. It might even be possible to automatically determine the body height in a virtual reality application through an initialization phase. This would be a use case for future research.

The presented results showed that hand tracking using the (improved) RGB method has significantly higher tracking ranges (with usable distance well above arm's length range) as well as the ability to track more than just two hands at a time. This can be especially useful for hand recognition in larger tracking areas with multiple users. Thus, we have also shown that by inputting the user's real hand size, a tracking system based on a single image RGB camera can provide results that provide a tracking error that could allow direct virtual interactions, as well as significantly expand the tracking range for hands. This method could be superior to the presented commercial systems such as LeapMotion or Meta Quest for specific scenarios as colocated multi-user scenarios, where one also wants to track the hands of the other users.

Therefore, it would be interesting in future experiments to see the effects of such RGB tracking in multi-user VR scenarios and to find out how it can disrupt or improve hand tracking in such applications. Since our setup and use case focuses on a single camera for tracking, using a camera array to improve tracking of colocated users' hands would be an interesting extension for the future.

One limitation of the measurements in the experiments shown was that only the distance of the palm of the hand was measured. A more detailed analysis of the distance errors of all finger joints would be an interesting extension of these results.

With the successful implementation of the presented work, we can now create a colocated VR environment using hand tracking. This system allows us to track multiple hands and accurately position them in three-dimensional space within an acceptable margin of error, enabling seamless interaction for each user. However, the introduction of multi-user hand tracking presents a new challenge – the reliable assignment of recognized hands to virtual users. This is crucial for tailoring specific interactions to individual users. The forthcoming chapter will address this challenge and explore various approaches to solving it.

CHAPTER 6

Assignment of Tracked Hands in Colocated VR

The upcoming chapter will focus on our approach of assigning virtual hands to colocated virtual users. We will present various methods for calculating confidences that a specific hand belongs to a particular user. These methods are designed to accommodate different scenarios, utilizing either the distance or rotation of the hand to the user, bounding boxes of the user's reach area, or employing a machine learning approach for assignment confidence calculations.

To enhance assignment accuracy and explore initial steps towards performance improvements, we introduce algorithms that build upon the base methods. One algorithm dynamically selects the most suitable base method based on user formations, while another leverages historical cached results to predict future hand assignments, addressing temporary weaknesses in the assignment algorithm.

To determine the effectiveness of the proposed methods and improvement algorithms, we conducted an extensive evaluation using simulated colocation scenarios. Our analysis encompasses accuracy and performance requirements, providing insights into the usability of these methods in real-time scenarios. The chapter concludes with a recommended method for hand assignment and offers a future outlook on potential enhancements to the results.

6.1 Motivation

In a colocated VR scenario where each user utilizes their own hand tracking system, the assignment of tracked virtual hands is straightforward. Given that many tracking systems, such as LeapMotion, Meta Quest, or Vive Focus, are typically limited to tracking two hands at most [37], the assignment is made based on the tracking system that initially detected the hand. Figure 6.1 provides an illustration of this process.

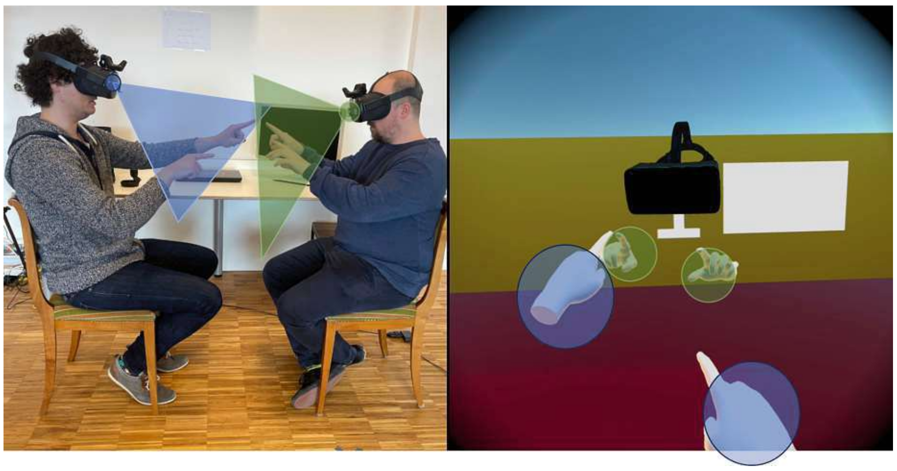


Figure 6.1: A colocated scenario involving two users, each equipped with their own hand tracking system. Virtual hands in the right image are assigned to the system that tracked the corresponding real hand in the left image.

With the introduction of hand tracking capable of simultaneously tracking more than two hands, as detailed in chapter 5, the assignment process becomes considerably more complex. This challenge is illustrated in Figure 6.2. In scenarios where a tracking system can detect more than two hands at once, it becomes impractical to assign virtual hands directly to the user equipped with the hand tracking system. To ensure consistent three-dimensional interactions, alternative methods must be explored to reliably assign virtual hands to the correct colocated virtual user.

Current research addressing this problem focuses either on image tracking [113][80] to extract additional information, such as overlapping tracking boxes or hand color, or is limited to two-dimensional images [59]. To our knowledge, there is currently no solution available that addresses this assignment problem in a three-dimensional virtual environment where only location and orientation information is available. This is especially difficult for hands in close proximity. Consequently, the objective of our work is to assign all virtual hands in 3D space to their respective users solely based on the location of the tracked hands relative to all present virtual users, and therefore enabling targeted interactions within the virtual environment.

In this chapter, we present an algorithm that computes the confidence of a virtual hand belonging to a specific user. We also address the challenging task of accurately assigning hands when they overlap in the virtual scene. To achieve this, we demonstrate various methods for computing this confidence and compare their effectiveness. In addition, we introduce a history-based algorithm that utilizes previous assignments and confidences to adjust future assignments (comparable to dead reckoning in multiplayer games [77]).

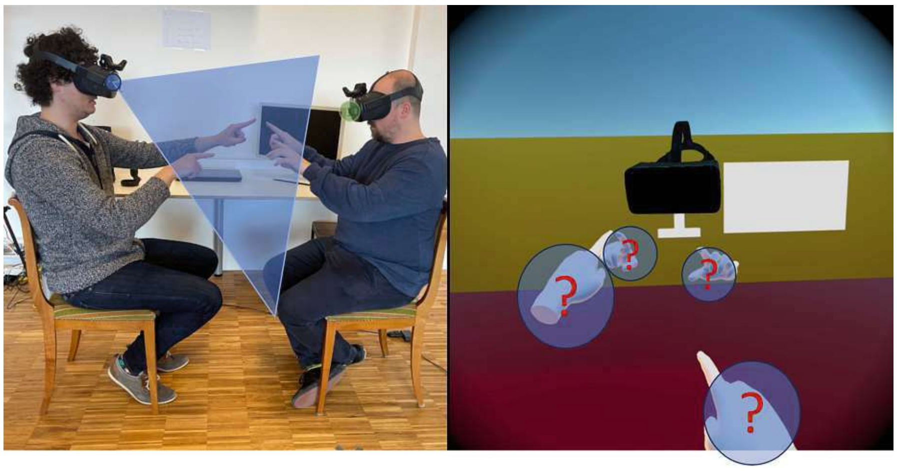


Figure 6.2: Two colocated users where only one user is equipped with a hand tracking system that is able to track more than two hands. Virtual hands on the right image cannot be reliably assigned to the correct user.

Finally, we evaluate our methods within a colocated virtual reality simulation, considering different user formations involving overlapping hand interactions.

Our evaluation focuses on assignment accuracy and the performance of seven method combinations to identify the most reliable and robust approach for assigning hands in all possible colocated virtual reality scenarios. We also assess the effectiveness of the history algorithm in improving assignment accuracy.

We aim to offer an algorithm for accurately assigning hands to virtual users without relying on additional image recognition methods, solely using the position and rotation of the hand and the user, which are provided by the used HMD and hand-tracking system. Additionally, we aim to solve the assignment problem that arises when tracked hands in the virtual scene are in close proximity, which is also particularly useful when a user's hands are obscured (for example by other body parts).

6.2 Estimation Methods

Our algorithm aims to enable the most reliable hand assignment to users, taking into account the unique challenges of multi-user colocation scenarios, such as close proximity of the users and overlapping hands. In traditional hand-tracking systems, it is often assumed that only the hands of the primary user will be tracked, as most systems are limited to tracking a maximum of two hands at a time [37][119][107]. However, in colocated scenarios, it is necessary to assign the hands of each user to their respective tracking systems.

A naive approach to hand assignment is to assign each virtual hand to the user closest to it. However, this approach can be error-prone in formations where users are close to each other and virtual hands are spatially intermixed. An example of such a scenario would be a surgical or assembly task in which several users perform hand interactions in a confined space. In our algorithm, we consider several algorithms, including distance and rotation, to compute a confidence factor for each virtual hand's assignment to a user.

In addition, we store the previous assignment of hands to users in a history cache to improve the accuracy of future calculations. By considering the history of previous hand assignments, our algorithm can make more reliable predictions about to which user a particular virtual hand belongs to.

6.2.1 Empirical Estimation Methods

We developed four approaches to determine the affiliation of a hand to a user, each of which computes a confidence factor p.

Distance: We calculate the euclidian distance of the hand to the user's head and assume that the closer the hand is, the more likely it is to belong to the user. This principle is illustrated in Figure 6.3. If the distance exceeds a threshold value, we no longer assume that the hand belongs to the user. The threshold was chosen so that it is slightly larger than an arm's length.

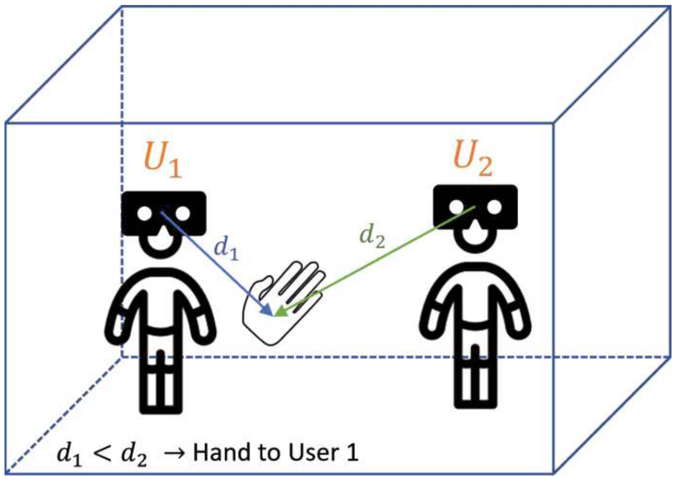


Figure 6.3: Distance method: The distance between the hand and the user is calculated. The hand is assigned to the user with the shortest distance.

To calculate the confidence values between 0 and 1, we use a logistic curve, as shown in Equation 6.1. We chose a distance range of [0.4m;1.3m] that results in $a = 10.22$ and $b = 0.85$. Figure 6.4 shows the resulting logistic curve. This method was selected based on its simplicity and anticipated high accuracy in numerous colocated scenarios, where users are sufficiently separated from each other.

$$p = 1 - \frac{1}{1 + e^{-a(x-b)}} \tag{6.1}$$

where:

a = the logistic growth rate
b = the x value of the functions mid point

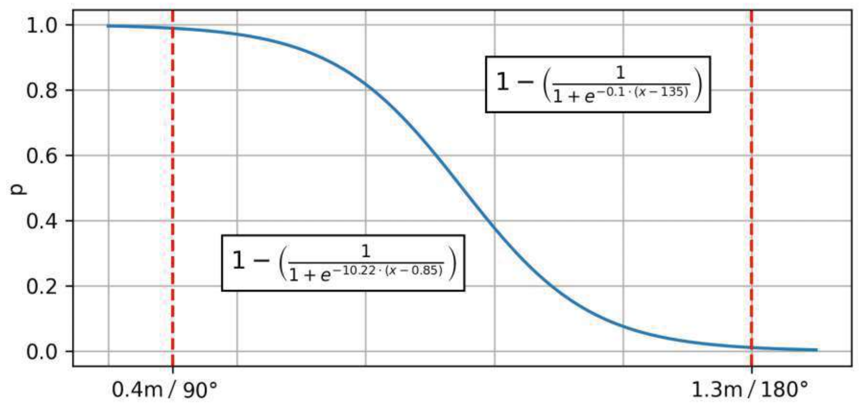


Figure 6.4: Logistic curve for distance (formula at bottom left)) and rotation (formula at bottom right) calculations. The X-axis represents the input position/rotation, while the Y-axis denotes the resulting confidence. The red lines indicate the selected threshold values.

Rotation: We calculate the angle between the forward direction of the hand (calculated via the connection between wrist position and position of the lowest joint of the middle finger) and the direction vector between the head of the user and the hand. We assume that a hand turned away from the user is more likely to belong to the user than a hand turned towards the user (which would require an uncomfortable hand position). This method is further illustrated in Figure 6.5.

We calculate p using the logistic curve in Equation 6.1 with values in the angle range [90°;180°], which results in the values $a = 0.1$ and $b = 135$. The logistic curve can be seen in Figure 6.4.

Prerecorded Area: In a preliminary setup, we asked the user to stretch out their arms and perform defined movements in front, to the side, and behind their body, resulting in a total of 459 points. A convex hull is then created from the points at performance, in which it is checked whether the user's hand is in this hull or not. Our focus was on capturing the essential movements rather than the precise number of points to ensure comprehensive coverage of all relevant sides of the body. Such a convex hull can be seen in Figure 6.6. The fundamental idea is that although this approach is more complex than using the distance method, it offers a more accurate representation of the radius of

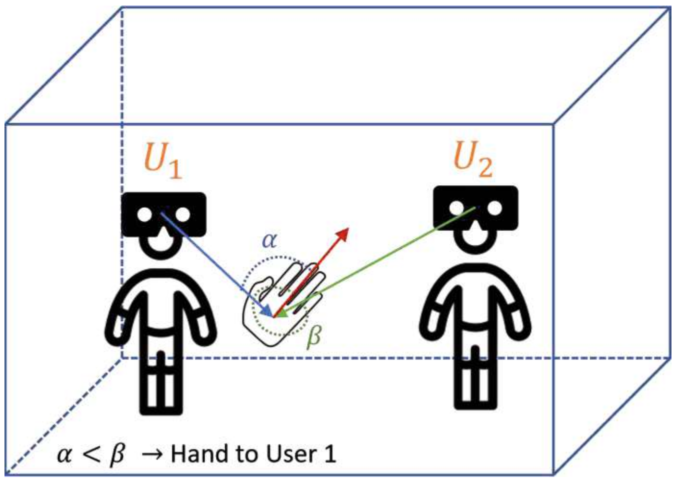


Figure 6.5: Rotation method: The angle between the forward vector of the hand and the direction vector from hand to user is calculated. The hand is assigned to the user with the smaller angle.

the hand. For instance, it acknowledges the limitation that hands cannot be positioned as far behind the user as they can be in front of the user. If there are several hands in this area at the same time, you can additionally use the 'Distance' method for better differentiation (as suggested in section 6.2.3).

6.2.2 Hand Assignment with Machine Learning Agents

In this approach, we used Unity3D's machine learning agents [45][12] for hand assignment using reinforcement learning. Unity's Machine Learning Agents (ML-Agents) is a toolkit that allows developers to integrate machine learning algorithms with Unity's game development platform. It enables the creation of intelligent agents within Unity environments, allowing them to learn and adapt to tasks through machine learning techniques. The technology and algorithms behind Unity ML-Agents involve reinforcement learning, a type of machine learning in which an agent learns to make decisions by interacting with an environment [56][1]. The implementations of these algorithms are based on PyTorch, an open-source Python library focused on machine learning [2].

During training, an agent received input data, including hand and user locations, and received rewards for correct assignments. Hand and head movements were recorded in advance and played back during the learning process to obtain the most realistic learning data possible. To enrich the learning experience, we introduced random user placements within a radius of 2 meters, with locations changing every frame. Additionally, we enforced specific scenarios with overlapping hands when users were close to train the

[1]Unity ML Agents: `https://github.com/Unity-Technologies/ml-agents(Accessed: 2024-01-25)`

[2]PyTorch: `https://pytorch.org/(Accessed:2024-01-25)`

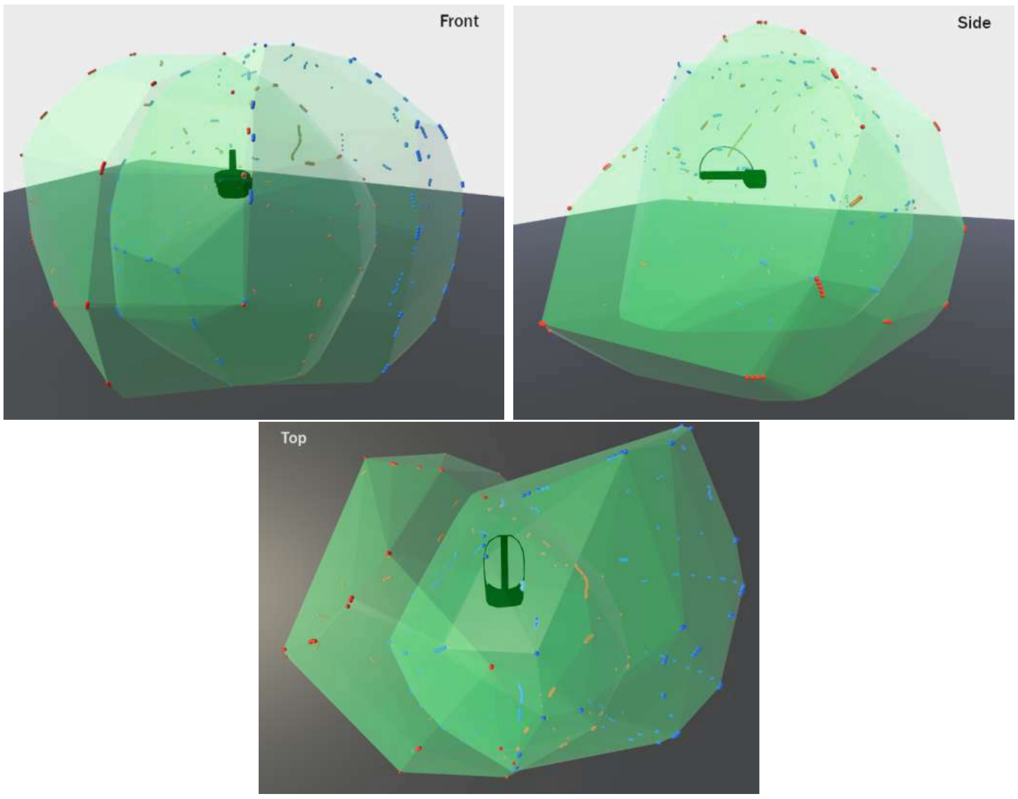


Figure 6.6: Convex Hulls generated from user's VR controller inputs. The red points correspond to recordings from the right hand, while the blue points represent recordings from the left hand. The resulting convex hulls are depicted in green.

agent for challenges. For varying user counts, we created dedicated agents (e.g., one for two users, one for three users) to handle increased decision complexity. Training involved 1.5 million steps, and reward progression can be seen in Figure 6.7, showing smoothed reward curves. In our tests, we trained up to five simultaneous users (which was sufficient for our comparative analysis), but we could easily extend to more simultaneous users.

This training data served as training input for Unity's ML system, enabling the creation of an agent capable of runtime hand assignment using the same type of input data (positions, rotations). In the assignment process, each visible hand in the scene is assigned to a specific user using its own agent, depending on the number of concurrent users. Agents were created for 2-5 concurrent users, but more concurrent user agents can be easily added. A decision is made for hand assignment in each rendering frame. We anticipate that this method can leverage multiple inputs during the learning process to autonomously make real-time decisions. Consequently, we expect it to offer broader coverage of various colocated situations.

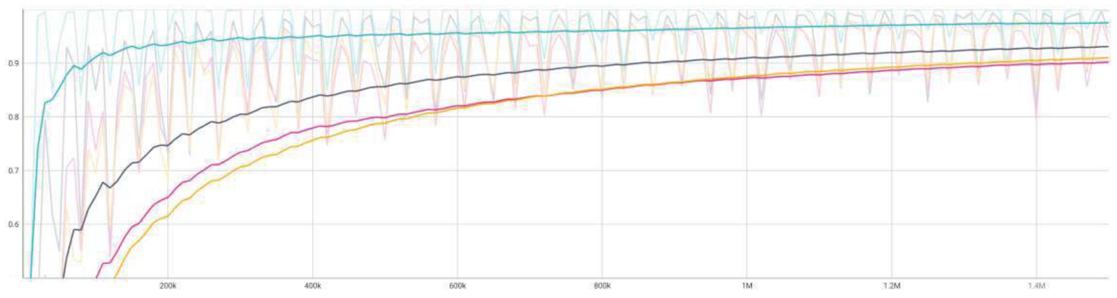


Figure 6.7: Reward development in Unity3D for ML training. The blue line corresponds to training with 2 users, the grey line represents 3 users, the pink line denotes 4 users, and the yellow line represents 5 users. The lines are smoothed for clarity, while the transparent lines in the background display the unsmoothed results. The variance in unsmoothed rewards may arise from randomized input data.

6.2.3 Dynamic Method Selection

This method combines the other methods to dynamically determine which method to use in the current situation. The idea is that if all users to be tested are far apart, the distance methods can give a very reliable indication of the hand's affiliation. However, if at least two users are spatially very close to each other, the assignment is no longer unambiguous, and therefore a more reliable method is used when hands overlap. Figure 6.8 illustrates such a possible dynamic selection.

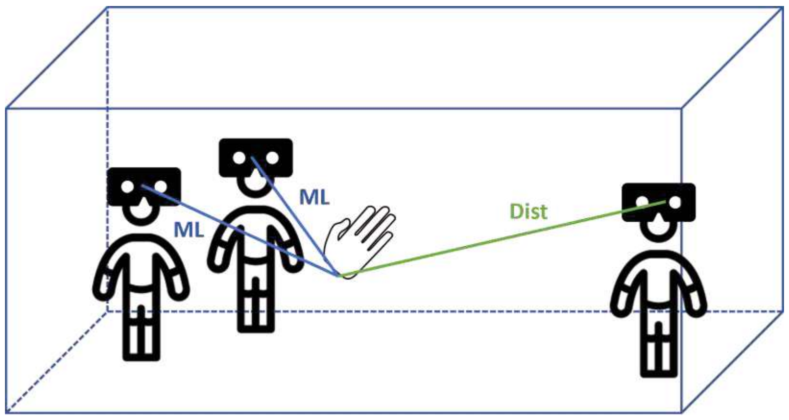


Figure 6.8: An example of a dynamic selection of hand assignment methods for a hand. The two left users and the hand are in close proximity. The complex 'Machine Learning' algorithm is used here. The user on the right is further away, which is why the simpler 'Distance' algorithm is used.

The idea is that computationally intensive methods (such as 'Machine Learning') are only used in situations where simpler methods are no longer sufficient, hence the computation time compared to using the complex algorithm alone. The effectiveness of these savings is examined in section 6.4.3. In our test implementations, we use the combination of 'Prerecorded Area' when far away and 'Distance' when near. We use 'Distance' method in close proximity, since 'Prerecorded Area' alone cannot make a distinction for multiple hands in the same area and thus can still expect a comparably low computation time. Additionally we use the combination of 'Distance' when far away and 'Machine Learning' when near as we assume that the 'Machine Learning' method provides better results than the 'Distance' method in close proximity, but also has a higher computation time. Therefore, the 'Distance' method is used for the other proximities. This should improve the overall computation time compared to the use of 'Machine Learning' alone. As threshold distance, we preliminarily tested assignment accuracy and determined a near distance of 0.8m as effective.

6.2.4 Assignment History

In a naive mapping of a hand to a user, the probability that a hand belongs to a particular user is computed independently for each hand and user in isolation. However, the previous assignment of a hand may have a significant impact on future assignments. To address this issue, we investigate whether incorporating historical information can improve the assignment of hands. Our thought was that if we assign hands correctly in the past with a high confidence, for example when users are further apart, we can catch the problems in close scenarios when hands overlap. The idea is that if a hand has been confidently assigned to a user before, then it is likely to belong to that user later. This approach is similar to the basic idea of dead reckoning, where for example in multiplayer games previous player network states are used to predict the next states to counteract network latency [77].

In our implementation, we cache the calculated confidences of each hand for each user for each frame. We limit the number of entries to 500 for memory and runtime efficiency, with new entries replacing the oldest ones. From all cached entries, we compute the mean confidence value (ranging from 0 to 1) and use it to calculate the total confidence. However, we only incorporate the history information if the value of the history exceeds a certain threshold. Pilot tests indicate that a confidence threshold of $p >= 0.9$ (90%) is appropriate.

Furthermore, we compute a weighting for the history so that high prior confidences are weighted more heavily in the overall computation, assuming that a frequently occurring high confidence in the past probably means a correct assignment. This should counteract low new confidences up to the point where they also occur frequently. As a result, the history confidence can be multiplied by a weighting factor, depending on the previous confidences. Another approach would be to weight newer confidences higher due to their actuality, which was not used in this work and would be an interesting comparison for future work.

To ensure that the maximum display of the raw input confidence is maximally doubled (which we chose to avoid over-weighting of the past confidences) and to ensure that the history algorithm returns to lower confidences after consecutive frames with low input raw confidences occur, a weighting factor of 2 was selected. This expected behavior is specifically anticipated in such a scenario. The confidence p of the history is calculated using the following formula:

$$p_h = \frac{\sum_{n=0}^{N} p_n}{N} * \frac{\sum_{n=0}^{N} p_n * w_f}{N_{max}} = \frac{w_f * (\sum_{n=0}^{N} p_n)^2}{N * N_{max}} \tag{6.2}$$

where:

N	= number of entries in the history
N_{max}	= maximum number of entries in the history, set to 500 in our implementation
w_f	= maximum weight, set to 2 in our implementation

Therefore, a reliable previous assignment of a hand can also be critical for future calculations if an assignment is no longer straightforward. For instance, in dynamic scenarios where users start far apart but later converge to interact with the same virtual object, relying solely on distance for hand assignment becomes less effective. In such situations, the use of historical data can help to make more accurate hand assignments. This underscores the value of incorporating historical information in dynamic contexts.

6.3 Experiment

To assess the effectiveness of various hand assignment estimation methods and examine the impact of the history algorithm, we developed an evaluation framework that simulates diverse test scenarios within a colocated VR environment. Each scenario comprises specific user formations and whether users move or are positioned stationary. The objective of this evaluation is to provide comparable scenarios for different combinations of assignment methods and to conduct a quantitative analysis of the accuracy and performance of the different methods. Since we want to use hand assignment in a colocated VR environment (which we created in chapter 4) in which a hand recognition system can recognize more than two hands (as established in chapter 5), it is necessary to use it in a real-time environment. Therefore, in this experiment, we also want to find out which methods are best suited in a real-time application regarding the performance and accuracy.

6.3.1 Experimental Design and Evaluation

We created a simulated environment that maps a colocated VR application, with 2-5 simulated users as needed. For each user, hand movements were recorded by us beforehand over a period of 1500 consecutive frames, representing an activity such as assembly work (using a screwdriver) or typing on a keyboard. These recordings can then be played back for different test scenarios, making these scenarios similar in their hand movements and thus comparable. Each simulated user had different recordings to cover different tracking scenarios. To maintain repeatability and comparability, it was ensured that the recordings in the evaluation are always assigned to the same virtual user and that the virtual hands thus also have the same recorded relative distances and rotations to this user.

Each test scenario is defined by the following factors:

- **Formation:** A predefined formation of how the virtual users are positioned with respect to each other in the virtual space and at different distances.
- **Method:** An assignment method, as described in section 6.2.
- **History:** Whether history is used as in section 6.2.4.

Specifically, we use the methods described in section 6.2 in isolation (Distance **(D)**, Rotation **(R)**, Prerecorded Area **(PA)**, and Machine Learning **(ML)**). In addition, we use a combination of Prerecorded Area and Distance **(D-PA-C)** to see if this combination improves the individual results. The selection of the methods was made in a preliminary evaluation run. Two constellations were used for the dynamic method: Area Prerecorded as a base method with Distance as an additional method when users are close to each other **(PA-D-DYN)**; and Distance as a base method and Machine Learning when users are close to each other **(D-ML-DYN)**. The idea is to use a simple algorithm for straightforward scenarios (i.e. hands are further away from the user) and switch to a more complex, potentially slower one when necessary, such as in close proximity when hands are overlapping. This approach aims to improve the average performance while maintaining good results. Therefore, a total of seven method constellations were evaluated. We use the abbreviations mentioned above for the methods and their combinations in the analysis. 'C' stands for combination of methods and 'DYN' for a dynamic selection between two methods. In the dynamic selection, the method in the first place is the one that is used at further distances, and the second is the one for close proximity.

In the simulation, we assume that all virtual hands are detected by a hand detection system, but remain unassigned to users. With ground truth information about hand assignment (which we have through the recordings and the controlled simulation setup), we can compare assignment results. We calculate an **accuracy ratio**, representing the percentage of accurate assignments per frame during the evaluation. A ratio of 1 signifies perfect assignment accuracy, serving as a key metric for method comparison in our study.

In addition to assessing assignment accuracy, we evaluate the performance (in ms) of each method to determine its applicability in a real-time scenario and the computational time required. To find the overall performance per frame, we sum up the measured times for all frames and divide by the total frames collected. Anticipating decreased performance with more users, we calculate an adjusted performance per user by dividing the frame's runtime by the number of concurrent users, as elaborated in section 6.4.3.

The evaluation covers a total of 73 formations, comprising 14 formations with 2 users, 16 with 3 users, 20 with 4 users, and 23 with 5 users. We aimed to represent diverse formations where users stood both farther apart and close together, resulting in overlapping hands. Some of those formations can be seen in Figure 6.9. Variation in the number of formations is attributed to the increasing number of concurrent users, which in turn leads to a greater diversity of potential arrangements. The following example can illustrate this again: If we have 2 simultaneous users standing facing each other they can either stand close to each other or far away from each other, resulting in two formations. If we now take four simultaneous users, they can face each other (in a square) too. However, they can all stand close together, only some of them (two stand close together and the other two further apart), or all stand far apart. This creates more than two possible formations.

Additionally, we selected formations in which users remained fixed in one place and dynamic formations where users moved along a fixed path, representing stationary and moving users in colocated scenarios. Each of the seven methods mentioned above was applied to each formation for the duration of 1500 frames to determine the percentage of correctly assigned hands. This evaluation was carried out with and without applying the history algorithm, resulting in 14 conditions (7 method constellations with and without applying the history algorithm). In total, we obtained 1022 results for comparative analysis (for performance and accuracy).

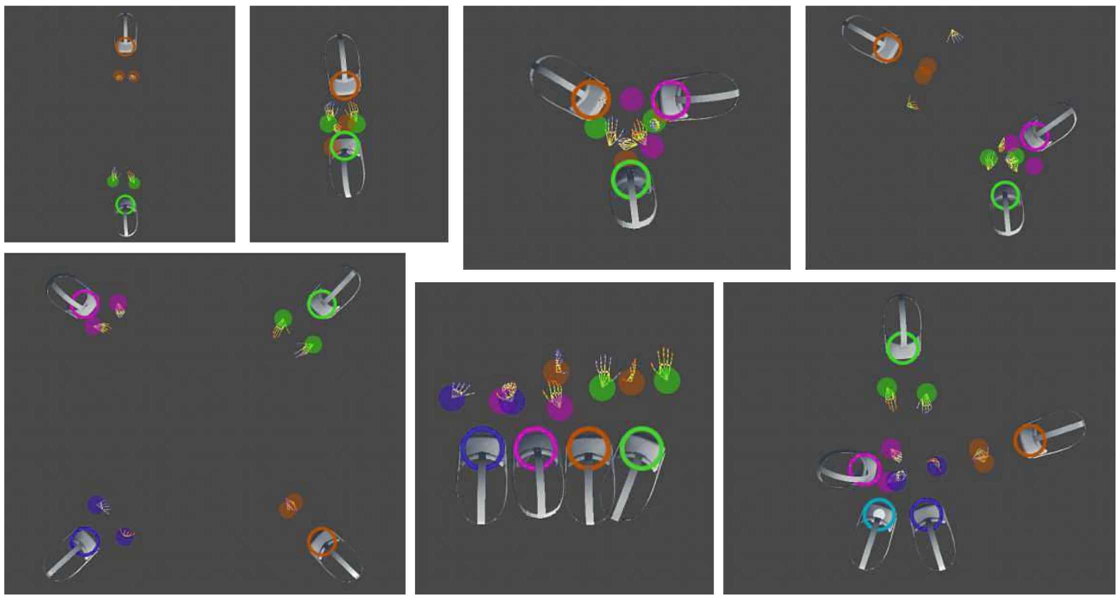


Figure 6.9: Some exemplary formations of the evaluation with two to five simultaneous users. Formations include near and far proximities to cover diverse and difficult assignment scenarios.

6.3.2 Setup

The evaluation was performed on a Windows notebook equipped with an Intel Core i7-12700H CPU and an NVIDIA GeForce RTX 3070 Ti Laptop GPU. We utilized Unity3D 2021.3.9 as the software platform, coupled with the 'EasyHand' framework for Unity that we presented in chapter 3. This framework served as a layer for hand tracking, and was responsible for recording tracked hands in a preliminary setup, which was done by us, playing back these recordings, and rendering the hands during the evaluation.

By conducting these evaluations, we gain insight into the accuracy of hand assignment and the computational efficiency of the methods, ensuring their feasibility in real-time applications.

6.4 Results and Discussion

We conducted an analysis of correct assignment ratio and computational efficiency to determine the optimal method for a real colocated VR scenario. Firstly, we evaluated the effectiveness of our history algorithm in improving hand assignment results. In addition, we identified the method or combination of methods that yielded the best results. To evaluate the algorithm and methods, we measured the accuracy of hand assignment for each frame, allowing us to calculate an accuracy ratio for each combination of method, formation, and utilization of the history algorithm. This resulted in a total of 1022 values, each representing the percentage of correctly assigned hands throughout the evaluation.

The second metric we examined was the performance of the algorithm. As our objective is to use this estimation method in real-time colocated VR applications, performance is a critical factor in determining the method's feasibility. The performance was measured in milliseconds per frame for each run.

Based on our implementation and the complexity of the methods, we formulated the following hypotheses for our expected results:

- **H1**: The history algorithm significantly improves hand assignments for all assignment methods.
- **H2**: The machine learning algorithm (ML) is significant more accurate than the distance (D), rotation (R) or Prerecorded Area (PA) algorithm.
- **H3**: The machine learning algorithm (ML) alone is significant slower than all other individual methods (D, R and PA).
- **H4**: A dynamic combination of the machine learning algorithm (ML) with the distance algorithm (D) can yield results with an accuracy comparable to that of machine learning alone, while also achieving a better performance.

By analyzing these metrics and testing our hypotheses, we aim to provide valuable insights into the performance and suitability of different methods for real-time colocated VR scenarios. To select the appropriate test for the comparative analysis in the subsequent sections, we performed a Shapiro-Wilk normality test [102] on each analysis. The test results suggested that the examined data did not adhere to a normal distribution ($p < 0.001$), leading to the use of the analysis methods mentioned in the following sections. We presume that the non-normal distribution might be attributed to the varying complexity of different methods, leading to distinct runtimes and diverse accuracy distributions across different scenarios.

6.4.1 History Algorithm Effectiveness

To assess the impact of the history algorithm, we analyzed all available test results, resulting in a sample size of 1022 entries. Half of the entries ($n = 511$) were executed with the history algorithm enabled, while the other half was executed without it. The objective was to determine whether the use of the history algorithm leads to a significant improvement in the accuracy of the assignment of the hand. The results are visualized in Figure 6.10.

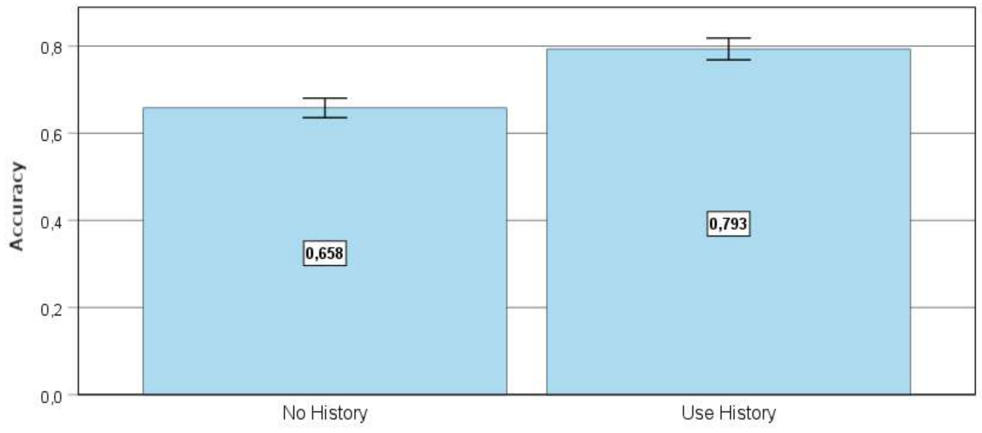


Figure 6.10: Accumulated accuracy across all methods with and without the history algorithm. The utilization of the history algorithm leads to an overall higher level of accuracy.

Since all methods, and thus also poor assignment accuracy of some methods, were included in the analysis, we do not consider the exact assignment accuracy of the history algorithm here, but only whether there is a significant improvement. A detailed assignment accuracy of the methods is shown in section 6.4.2.

Without employing the history algorithm, an average accuracy of **0.658** ($\sigma = 0.26$) was observed across all data. However, when the history algorithm was included, the accuracy increased to **0.793** ($\sigma = 0.29$). We conducted an Independent-Samples Mann-Whitney U test [79] for comparative analysis. The resulting p-value ($p < 0.001$; $U = 79317$), together with a rank-biserial correlation coefficient of $r = 0.333$ (indicating a positive effect size), demonstrate significant differences with a notable improvement in hand assignment accuracy when using the history algorithm. These findings confirm **H1**, as the utilization of our history algorithm led to significantly enhanced assignment results, showcasing an average improvement of **20%** in our evaluations.

6.4.2 Assignment Accuracy

In this analysis, the methods listed in section 6.3.1 were compared based on their accuracy. Since the previous section demonstrated the superior performance of the history algorithm, we focused on runs where the history algorithm was applied to achieve the highest possible accuracy in our evaluations. This resulted in a sample size of 511 entries (consequently half as much as for the analysis in section 6.4.1), corresponding to N=73 per method with 7 applied methods.

Initially, we calculated the means and standard deviations (σ) of the accuracy for all methods, which can be found in detail in Table 6.1. From the table, it is evident that the methods that include machine learning exhibit the highest accuracy with the lowest standard deviation. In contrast, the rotation method yielded the lowest average accuracy, with approximately half of the assigned hands being incorrect on average. The corresponding bar graph can be seen in Figure 6.11.

Table 6.1: Mean and standard deviation of accuracy of the methods

Method	**Mean**	**SD** σ
Machine Learning **(ML)**	0.987	0.033
Distance **(D)**	0.729	0.31
Rotation **(R)**	0.584	0.334
Prerecorded Area **(PA)**	0.738	0.265
Prerecorded Area Distance Combination **(D-PA-C)**	0.782	0.291
Dynamic Prerecorded Area Distance **(PA-D-DYN)**	0.744	0.30
Dynamic Distance Machine Learning **(D-ML-DYN)**	0.989	0.033

We conducted an Independent-Samples Kruskal-Wallis test [51] to assess the differences in accuracy among all methods. The analysis yielded a significant result ($p<0.001$; $H=92.9$; $df=6$) and a medium effect size, as measured by η-squared ($\eta^2 = 0.172$), indicating substantial differences in the distribution of accuracy across the methods.

To further investigate these differences, we performed a post-hoc test for pairwise comparisons between the methods. We applied Bonferroni correction (with $k = 21$) to adjust the resulting p-values. Detailed p-values can be found in Table 6.2.

The rotation method exhibited significantly worse results in terms of accuracy compared to all other methods. The machine learning method demonstrated significantly better accuracy compared to the distance and rotation methods. Although significance in

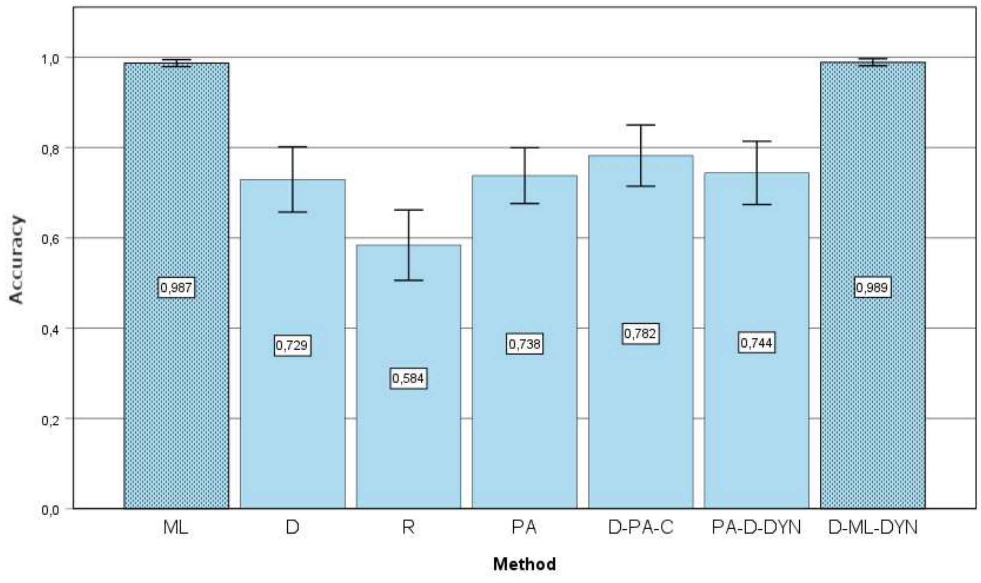


Figure 6.11: Mean accuracy results for the methods. The two most accurate methods are highlighted.

Table 6.2: Resulting p-values of the pairwise comparison of the methods after applying the Bonferroni correction. Significant results are marked in bold.

p	**ML**	**D**	**R**	**PA**	**D-PA-C**	**PA-D-DYN**
D	**.032**					
R	**<.001**	**.013**				
PA	.066	1	**.006**			
D-PA-C	.469	1	**<.001**	1		
PA-D-DYN	.141	1	**.002**	1	1	
D-ML-DYN	.424	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**

accuracy was initially observed between the dynamic methods before Bonferroni correction, it was no longer present after adjusting for multiple comparisons. However, it is worth noting that the machine learning method exhibited the least variance and the most consistent accuracy across all runs, as evidenced by its low standard deviation and consistent data distribution.

The dynamic combination of distance and machine learning shows an even more favorable outcome, showing significant differences compared to all other methods except for the standalone Machine Learning method. It also displayed high consistency with a low standard deviation ($SD = 0.033$).

With an accuracy of **99%**, both the machine learning algorithm and the dynamic combination of machine learning and distance emerged as significantly superior algorithms for hand assignments. Therefore, these findings support Hypothesis **H2**, that the machine learning algorithm yields the best accuracy among the single used algorithms.

The remaining methods fell within a similar range of average accuracy, ranging from **72%** to **78%**, and did not produce significant differences from each other. This suggests that the choice of method in this range should be based on other factors, such as performance.

6.4.3 Performance Results

In our final decision regarding the optimal hand assignment algorithm, it is crucial to consider the performance to assess the feasibility of the method in a real-time scenario. Since we have already established in section 6.4.1 that the history algorithm significantly improves hand assignments, we also need to investigate if it introduces a substantial decrease in performance. In addition to accuracy, performance plays a vital role in selecting the best method.

To obtain the final performance for the most effective hand assignment method, we calculated and compared the performance for each method when the history algorithm was employed. As the allocation methods are applied to active hands and simulated users in each frame, the computational load increases with more users and hands. To account for this, we used a user-adjusted performance per computation frame for the analysis by dividing the measured frame runtime by the number of users.

By analyzing the performance results, we can gain insight into the computational performance of each method and identify any significant differences. These findings will aid in selecting the most suitable algorithm for hand assignment, taking into account both accuracy and performance.

History performance To assess the impact of the history algorithm on performance, we evaluated performance across all available runs, resulting in a sample size of 1022, evenly split between runs with and without the history algorithm.

When using the history algorithm, we observed an average user-adjusted performance of **0.465 ms** ($\sigma = 0.22$). In contrast, the average user-adjusted performance without the history algorithm was **0.147 ms** ($\sigma = 0.24$). It is important to note that these mean values do not solely represent the performance of the algorithm, as they incorporate the values from different assignment methods. However, they indicate that the history algorithm generally results in a poorer performance.

A Shapiro-Wilk normality test confirmed that the data did not follow a normal distribution ($p<0.001$). Therefore, we conducted an independent-samples Mann-Whitney U test and calculated the rank-biserial correlation coefficient to determine the effect size. The test yielded a significant result ($p < 0.001$; $U = 40603$), and the coefficient (*r=0.689*) indicated a positive effect size, demonstrating significant differences between using and not using the history algorithm.

Table 6.3: Mean performance of methods (in milliseconds) based on number of users, including User-corrected times

Method	Performance (ms) ± σ				
	2 User	**3 User**	**4 User**	**5 User**	**User-corrected**
ML	1.63 ± .75	2.37 ± .09	3.58 ± .14	4.09 ± .27	**.83 ± .17**
D	.55 ± .03	1.01 ± .04	1.49 ± .08	1.88 ± .15	**.35 ± .04**
R	.55 ± .01	1.03 ± .03	1.60 ± .12	2.03 ± .18	**.37 ± .06**
PA	.54 ± .02	1.01 ± .03	1.47 ± .07	1.88 ± .16	**.35 ± .05**
D-PA-C	.57 ± .05	1.02 ± .04	1.52 ± .10	1.95 ± .16	**.36 ± .05**
PA-D-DYN	.55 ± .03	1.03 ± .03	1.54 ± .07	1.97 ± .18	**.36 ± .05**
D-ML-DYN	1.08 ± .48	1.88 ± .66	2.81 ± 1.02	3.51 ± .94	**.65 ± .23**

Method performance Considering the significant improvement in the accuracy achieved by the methods, we proceeded to analyze the performance of each method when combined with the history algorithm. The mean performance values per number of users, along with the user-corrected value, are presented in Table 6.3 and are visually represented in Figure 6.12.

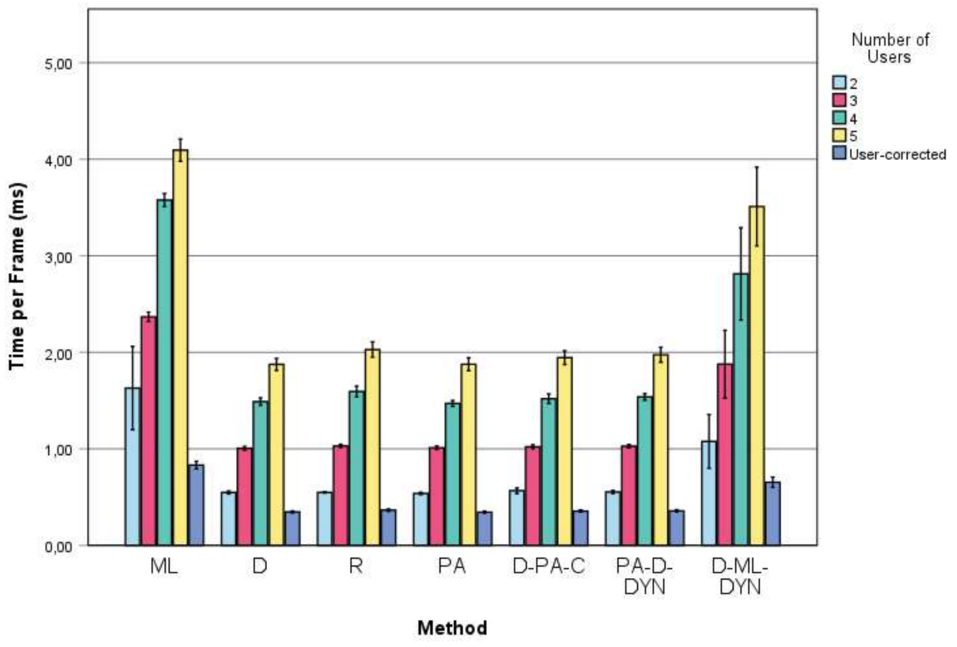


Figure 6.12: Mean performance results for the methods at various user counts. One bar illustrates the mean performance adjusted for the number of users.

Again, we performed a Shapiro-Wilk test to assess the normality of the data. The test yielded a p-value of less than 0.001 for all methods, indicating that the data is not normally distributed. Consequently, we conducted an Independent-Sample Kruskal-Wallis test once again. The analysis revealed significant differences among the user-corrected performances of the methods, with a p-value of less than 0.001 (H = 249.74; df = 6) and a positive effect size (calculated using η-squared) of $\eta^2 = 0.484$. To determine the specific differences, post-hoc pairwise comparisons were performed with Bonferroni correction (with k = 21).

The results indicate significant differences between the Machine Learning (ML) method and all other methods, as well as between the Dynamic Method with Machine Learning and Distance (D-ML-DYN) and all other methods, each with a p-value of less than 0.001. Since both of these methods involve Unity3D's Machine Learning Agents, it was expected that they would have better performance, as reinforced learning agents require more CPU time. These findings confirm our hypothesis **H3**.

However, it should be noted that there was also a significant difference between the two methods using machine learning (p = 0.047). This indicates that the dynamic combination, with a mean performance of 0.65ms, is significantly faster than the Machine Learning method alone, which has an average performance of **0.83ms**, by **27.7%**. When considering the results of section 6.4.2, hypothesis **H4** can also be confirmed. The dynamic combination of machine learning and distance algorithms exhibits a significantly better performance while achieving the best accuracy among all the applied methods.

No significant differences were observed among the remaining methods that do not utilize machine learning agents, suggesting similar performance for these methods. With an average performance of approximately **0.35ms**, these methods are **85.7%** faster than the dynamic method with machine learning and **137.1%** faster than the isolated machine learning method. In the next section, a more comprehensive discussion on the applicability of the methods based on these results, along with the previous findings, is presented.

6.4.4 Usability in Realtime Applications

The results demonstrate significant differences in the effectiveness of different algorithms for assigning recognized hands to users in colocated VR scenarios. As expected, the rotation method performs the worst, with approximately half of the virtual hands being incorrectly assigned. This outcome is logical since hands typically have a specific orientation to their owners, but can be freely rotated. The assignment of hands becomes challenging when two users have similar orientations in the scene, resulting in ambiguous assignments. Although a combination of the rotation method with other methods could potentially yield improvements, preliminary unstructured tests did not reveal any visible enhancement over other methods. Furthermore, since the rotation method does not offer significantly better performance, it is not suitable for reliable hand assignment.

The distance-based methods, whether utilizing the distance or a prerecorded area, deliver similar results across all examined aspects. With an accuracy of approximately three out of four hands correctly assigned over all group sizes, the performance is better than that of the rotation method, but still insufficient to ensure a reliable assignment. Contrary to our assumptions, the precise definition of the hand's area of action (prerecorded area) does not yield significantly better results compared to the pure distance method, which essentially represents a radius around the user. Additionally, the combination of these two methods did not confirm the notion that they could mutually support each other. Given similar accuracy and performance, the distance algorithm is preferable to the prerecorded area algorithm in terms of simplicity.

Taking into account Figure 6.13, it becomes evident again why the distance methods do not exhibit high overall accuracy. When examining user information where users are far apart, the assignment of hands can be achieved with a high confidence of success. However, if users are standing close to each other and their hands overlap, assigning hands based solely on distance becomes significantly more challenging. Consequently, distance methods are more suitable for scenarios where users in colocated spaces are not in close proximity to each other.

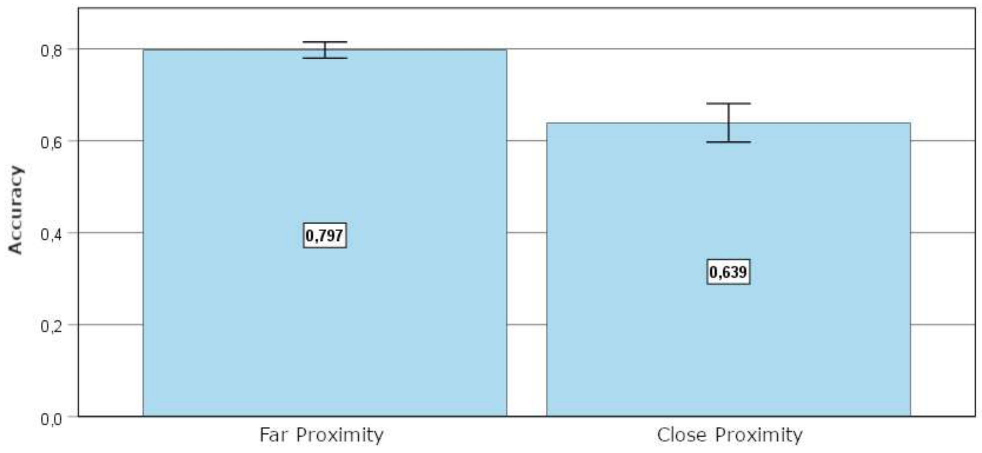


Figure 6.13: Effect of user proximity on the accuracy for the distance method. Reduced accuracy and increased variance can be seen when users are near each other. Figure 6.9 illustrates formations with different proximities.

When examining the methods involving machine learning, the results demonstrate a significantly improved assignment accuracy. Prior reinforcement learning of an AI agent was crucial in effectively determining the assignment factors. However, the one-time learning effort is no longer necessary in subsequent applications, highlighting its ability to produce good results. With an accuracy of approximately 99%, these machine learning methods exhibit the highest accuracy in hand assignment, regardless of the proximity of users to each other. This distinguishes them clearly from distance methods and scores an even higher accuracy than existing algorithms [63].

Nonetheless, as expected, the machine learning methods also have the lowest performance due to the computational complexity involved, with a user-corrected performance ranging from 0.65 to 0.83 ms. Performance decreases linearly with the number of users and hands present in the scene. Its applicability is thus contingent on the number of users, as well as the complexity and rendering demands of the virtual VR scene. However, the performance remains sufficiently high to allow usage in real-time applications.

The dynamic combination of machine learning and distance algorithms proves to be the optimal approach among the presented methods. The results indicate that the weaknesses of the distance method in assigning hands to users standing close to each other (see Figure 6.13) can be effectively compensated for by the machine learning agent. As a result, a significantly improved performance (of 21%) can be achieved while maintaining a high accuracy. This combination is particularly suitable for complex applications with high computational complexity. When users are consistently in close proximity to each other, the worst-case performance is comparable to that of using machine learning alone. On the basis of these findings, this combination is recommended for real-time applications.

Finally, the results highlight the significant improvement in the accuracy of hand assignments achieved by the history algorithm we developed. As expected, applying the algorithm also leads to a poorer performance, however, remaining at a reasonable level. The 20% improvement in assignment accuracy, coupled with an acceptable decrease in performance, establishes the algorithm as a substantial enhancement to the methods. Consequently, we incorporate it into our methods for optimal results.

6.5 Conclusion and Future Outlook

This research aimed to present and compare algorithms for assigning tracked hands to virtual users in a colocated virtual environment, using limited information from the tracking system. Unlike other solutions that rely on image detection information [113][59][63][132], we focused on utilizing the location of the hand in the virtual scene. We introduced and evaluated various methods for determining hand assignments, including a history algorithm that enhances assignment robustness by leveraging past assignment information.

Our evaluation revealed that assignment methods employing prelearned AI agents achieved the highest accuracy with the lowest variance. Specifically, the dynamic combination of an AI agent for close user proximity and overlapping hands, combined with a simple distance calculation for greater distances, improved the performance while maintaining a high accuracy of 99%. This dynamic combination effectively addressed the challenge of assigning overlapping hands. Although other methods exhibited better general performance, their accuracy fell short of being suitable for real colocated VR scenarios, where every fourth hand would then be assigned incorrectly. In scenarios involving hand tracking systems for tracking multiple hands, such as those utilizing camera tracking meshes, our solution provides developers with a high-accuracy and real-time hand assignment tool.

The computational load in a colocated VR environment increases with each additional user, which leads us to anticipate that this approach will eventually be constrained by the computational power of the system, ultimately limiting the maximum number of concurrent users before the application becomes impractical. Future applications should consider investigating the limitations concerning the maximum number of users and hands that can be simultaneously present in the virtual scene. Furthermore, it would be worthwhile to explore potential enhancements to the algorithm's performance. For instance, enhancing the performance of hand assignments can be achieved by optimizing the algorithm's execution frequency. This can be accomplished by avoiding the execution of the algorithm during each frame. For example, a proximity observer could rerun the algorithm every time the proximity of two users changes. This way the performance could be greatly improved. An avenue to improve the accuracy of assignment is to investigate the integration of image detection cues (as done in other work in the two-dimensional domain [80]), such as hand color, shape, and classification, to increase both accuracy and performance in hand assignment. Such explorations have the potential to significantly advance the effectiveness and efficiency of hand assignment estimation techniques.

CHAPTER 7

Conclusion

The focus of this dissertation and the research presented herein centered on the utilization of hand tracking technology, particularly in scenarios involving multiple colocated users. We began by introducing 'EasyHand', our framework for the Unity3D engine, which seamlessly integrates rendering, interaction, and gestures from various hand tracking technologies [1]. This framework served as the foundation for subsequent research and was designed with ease of extensibility in mind.

Several problems were presented, for which we have introduced solutions, including the creation of colocated virtual reality environments, the simultaneous tracking of multiple hands within such environments to facilitate the sharing of tracked hand data, and the consistent assignment of tracked virtual hands to their respective users, allowing consistent assignments of interactions to the right user. These are now integrated into the current version of the 'EasyHand' framework, or can be added as a plugin (see section 3.5). The overarching objective is to empower users in colocated multi-user settings to engage in virtual interactions by sharing tracking data, overcoming tracking losses of hands in ones own hand recognition systems.

In the following chapter, we will provide a final summary of the research objectives and outcomes, followed by an outlook on potential future extensions and unresolved research questions.

7.1 Summary

With this dissertation, we have approached several challenges that occur in hand tracking and colocated VR scenarios, as outlined in chapter 1. In the following section we will refer to the fulfilled contributions that were mentioned in section 1.2.

[1]EasyHand Repository: `https://bitbucket.org/Densen90/easyhand`

Hand tracking unification. In chapter 3, we introduced the 'EasyHand' framework, which serves as a unification tool for visualization, interaction, and gestures across multiple hand tracking APIs. This framework enables developers to easily extend support for any hand tracking technology, allowing them to deploy software across various VR systems while maintaining consistent behavior. Additionally, 'EasyHand' includes capabilities for networked scenarios, supporting both non-shared and shared virtual environments. This fulfills the contribution **I**. By integrating existing standardization such as OpenXR hand tracking [32], 'EasyHand' expands its functionalities and offers a versatile solution for hand tracking in VR applications.

Creating colocated scenarios for SLAM-tracked headsets using only hands. In chapter 4, we introduced a novel method for creating colocated virtual environments for SLAM-tracked headsets, using only tracked hands as synchronization anchors, hence affirming contribution **II**. This approach surpassed existing methods in synchronization accuracy, which often rely on additional AR markers or manual positioning of the headsets. Importantly, our method is headset-agnostic, as it relies solely on hand tracking and is not limited to a specific headset model. Unlike solutions such as the Vive Focus 3, which require sharing internal environment mapping data between headsets to create colocated environments [2], our approach offers greater flexibility and interoperability.

Tracking more than two hands in three-dimensional environments. In chapter 5, we addressed the limitation of current state-of-the-art hand tracking systems, which typically track only two hands simultaneously. Our solution involved using 2.5-dimensional tracking data from the MediaPipe hand tracking system in a three-dimensional space, confirming contribution **III**. This approach enabled VR interactions with a much larger tracking range than existing systems. Through an extensive evaluation, we compared our method with existing state-of-the-art hand tracking systems, demonstrating both comparability and improvements. Importantly, our method not only enables tracking of the user's own hand but also hands of other users within the tracking frustum of the camera. This capability allows and for sharing of tracking data among users, enhancing consistency in interaction and counteracting tracking losses within colocated scenarios. Therefore we can offer the contribution **IV**.

Assigning virtual hands to users. In chapter 6, we addressed the challenge of hand assignment that arises with the ability to track more than two hands simultaneously. We proposed solutions to this problem by introducing algorithms designed to assign virtual hands to the correct user. Through evaluation, we demonstrated that we can achieve this assignment with 99% accuracy in real-time scenarios, without the need for additional camera recognition or hardware. By solely utilizing positional data from hands and users, we enable correct and consistent interactions tailored to the respective user within colocated virtual environments. This finally fulfills contribution **V**.

[2]VIVE Environment Mapping: `https://business.vive.com/us/solutions/vive-location-based-software-suite/` (Accessed: 2024-02-08)

In summary, with 'EasyHand' we have introduced a comprehensive system capable of creating colocated VR scenarios, tracking both the user's own hands and those of other users within shared environments, and accurately assigning these hands to the correct virtual user. This framework enables the sharing of hand tracking data among users, facilitating tracking and interaction with the virtual environment even when hands are obscured, as long as at least one system can detect the hand. Importantly, all these functionalities are achieved solely through hand tracking, without the need for additional external sensors or hardware. This not only ensures affordability but also mobility and flexibility (as there is no room preparation required), as well as ease of use for both developers and end-users alike. With these results, we have achieved our overarching goal of providing ways for hand recognition systems to support each other in colocated VR scenarios. Thus, we see this work as an important improvement in the field of hand recognition and interaction in colocated multi-user VR scenarios.

7.2 Open questions

The research presented in this dissertation demonstrates the utility of hand-tracking technology in creating colocated multi-user environments and improving the consistency of interactions within such environments through multi-user hand tracking. Additionally, the dissertation addresses the challenge of assigning virtual hands to users in such scenarios. However, despite these advancements, several open questions remain, and opportunities for future research come up.

Usability and limitations in real user scenarios. Usability and limitations in real user scenarios are crucial aspects that warrant further investigation. While many evaluations and experiments in this work utilized simulated data or involved a limited number of testers[3], future research could benefit greatly from extensive user tests. We believe that our results are applicable to use in a real user scenario, as our simulations and tests have been designed to be as close as possible to such scenarios. Nonetheless, such tests would provide valuable additional insights into the effectiveness and precision of sharing tracked hand data among users, and furthermore, the use of our methods in real colocated scenarios for the case of hand tracking loss is theoretically possible, but a practical application in those cases is still pending. In addition, effects such as the camera moving instead of the hands in the evaluation in chapter 5 can be measured and analyzed. Lastly, they could help identify limitations of the proposed methods and systems, such as the maximum number of users supported depending on runtime constraints, the scalability of the methods for colocating multiple users, and the impact of sharing hand data on computational load. Addressing these aspects would contribute significantly to the practical applicability and refinement of the proposed solutions.

[3]Due to the COVID-19 pandemic and limited availability of potential participants.

Improving multi-user hand tracking. In our method, we utilized the MediaPipe framework to detect more than two hands simultaneously. While effective, this approach relies solely on RGB cameras and may lack some pose accuracy compared to systems with additional sensors, such as the LeapMotion sensor. We decided to use RGB cameras because of the lower price and better availability. Nonetheless, it would be beneficial to explore hand tracking systems equipped with improved sensor technology for higher accuracy and make them capable of tracking more than two hands. Such advancements could enhance the positioning of virtual hands, thereby improving hand ownership and facilitating the desired sharing of hand data among users. Integration of such technology into state-of-the-art HMDs like the Meta Quest or VIVE Focus holds promise for delivering a better multi-user hand tracking experience for end-users. Another interesting approach for the future would be to use multiple hand tracking systems per user. Each user has a 'primary' hand tracking system to recognize their own hand and a secondary one to assist other users if tracking is lost. In this way, the tracking workload could be divided between several tracking systems.

Continuous colocation with multi-hand tracking. The colocation method presented using hand tracking was a one-time process, which introduces the challenge of dependency on the drift of the HMD for further accuracy. To address this issue, a method could be developed to synchronize the virtual worlds of individual users continuously in multi-user hand tracking scenarios, every time, and as long as the hands are visible to the users. This approach could help mitigate potential drift. Additionally, sharing internal tracking data of the environment across systems could enhance colocation for multiple different systems. However, this would require approval from the system manufacturers and agreement on a standard for the data, making it unlikely to happen in the near future.

In summary, this dissertation has provided initial insights into calibrating multi-user environments, compensating for tracking loss and assignment of hands to users for colocated VR scenarios. While the focus was primarily on virtual reality, the methods presented are not limited to VR alone. Collaborative AR applications, such as multi-user surgery simulations or tabletop interactions where multiple users can interact simultaneously, stand to benefit from these findings as well. It is our hope that this work will have a significant impact on researchers and inspire further development and continuous improvement of hand recognition and multi-user interactivity in colocated scenarios. The natural input of hands makes this area particularly promising for future advancements.

Bibliography

[1] Diar Abdlkarim, Massimiliano Di Luca, Poppy Aves, Sang-Hoon Yeo, R. Chris Miall, Peter Holland, and Joseph M. Galea. A methodological framework to assess the accuracy of virtual reality hand-tracking systems: A case study with the oculus quest 2. *bioRxiv*, 2022.

[2] Christoph Anthes, Rubén Jesús García-Hernández, Markus Wiedemann, and Dieter Kranzlmuller. State of the art of virtual reality technology. *2016 IEEE Aerospace Conference*, pages 1–19, 2016.

[3] Daniel Bachmann, Frank Weichert, and Gerhard Rinkenauer. Evaluation of the leap motion controller as a new contact-free pointing device. *Sensors*, 15:214–233, 12 2014.

[4] Peter Bauer, Werner Lienhart, and Samuel Jost. Accuracy investigation of the pose determination of a vr system. *Sensors*, 21(5), 2021.

[5] Steve Benford, Chris Brown, Gail Reynard, and Chris Greenhalgh. Shared spaces: Transportation, artificiality, and spatiality. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work*, CSCW '96, page 77–86, New York, NY, USA, 1996. Association for Computing Machinery.

[6] Stina Bengtsson and Sofia Johansson. The meanings of social media use in everyday life: Filling empty slots, everyday transformations, and mood management. *Social Media + Society*, 8(4):20563051221130292, 2022.

[7] M. Borges, A. Symington, B. Coltin, T. Smith, and R. Ventura. Htc vive: Analysis and accuracy improvement. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2610–2615, 2018.

[8] Nicholas Brunetto, Nicola Fioraio, and Luigi Di Stefano. Interactive rgb-d slam on mobile devices. In C. V. Jawahar and Shiguang Shan, editors, *Computer Vision - ACCV 2014 Workshops*, pages 339–351, Cham, 2015. Springer International Publishing.

[9] Alvaro Parra Bustos, Tat-Jun Chin, Anders Eriksson, and Ian Reid. Visual SLAM: Why bundle adjust? In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, may 2019.

[10] Yunlong Che and Yue Qi. Detection-guided 3d hand tracking for mobile ar applications. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 386–392, 2021.

[11] Jun Cheng, Liyan Zhang, Qihong Chen, Xinrong Hu, and Jingcao Cai. A review of visual slam methods for autonomous driving vehicles. *Engineering Applications of Artificial Intelligence*, 114:104992, 2022.

[12] Andrew Cohen, Ervin Teng, Vincent-Pierre Berges, Ruo-Ping Dong, Hunter Henry, Marwan Mattar, Alexander Zook, and Sujoy Ganguly. On the use and misuse of abosrbing states in multi-agent reinforcement learning. *RL in Games Workshop AAAI 2022*, 2022.

[13] HTC Corporation. "customize 3d hand model". `https://hub.vive.com/storage/tracking/unity/model.html`, 2021. Accessed: 2023-10-06.

[14] HTC Corporation. "vive hand tracking sdk overview - supported hardware". `https://hub.vive.com/storage/tracking/overview/hardware.html`, 2021. Accessed: 2023-10-23.

[15] Connor DeFanti, Davi Geiger, and Daniele Panozzo. *Co-Located Augmented and Virtual Reality Systems*. PhD thesis, New York University, 2019.

[16] A. Del Bimbo, L. Landucci, and A. Valli. Multi-user natural interaction system based on real-time hand tracking and gesture recognition. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 55–58, 2006.

[17] Qicheng Ding, Jiexiong Ding, Jing Zhang, and Li Du. An attempt to relate dynamic tracking error to occurring situation based on additional rectilinear motion for five-axis machine tools. *Advances in Mechanical Engineering*, 12(10):1687814020967573, 2020.

[18] K. C. Dohse, Thomas Dohse, Jeremiah D. Still, and Derrick J. Parkhurst. Enhancing multi-user interaction with multi-touch tabletop displays using hand tracking. In *First International Conference on Advances in Computer-Human Interaction*, pages 297–302, 2008.

[19] Tobias Drey, Patrick Albus, Simon der Kinderen, Maximilian Milo, Thilo Segschneider, Linda Chanzab, Michael Rietzler, Tina Seufert, and Enrico Rukzio. Towards collaborative learning in virtual reality: A comparison of co-located symmetric and asymmetric pair-learning. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery.

[20] Ylva Ferstl, Rachel McDonnell, and Michael Neff. Evaluating study design and strategies for mitigating the impact of hand tracking loss. In *ACM Symposium on Applied Perception 2021*, SAP '21, New York, NY, USA, 2021. Association for Computing Machinery.

[21] Tiare Feuchtner. *Designing for Hand Ownership in Interaction with Virtual and Augmented Reality*. PhD thesis, Aarhus Universitet, Aarhus, 2018.

[22] Daniel Immanuel Fink, Johannes Zagermann, Harald Reiterer, and Hans-Christian Jetter. Re-locations: Augmenting personal and shared workspaces to support remote collaboration in incongruent spaces. *Proc. ACM Hum.-Comput. Interact.*, 6(ISS), nov 2022.

[23] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, 2006.

[24] Valentino Frati and Domenico Prattichizzo. Using kinect for hand tracking and rendering in wearable haptics. In *2011 IEEE World Haptics Conference*, pages 317–321, 2011.

[25] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[26] Milton Friedman. A correction. *Journal of the American Statistical Association*, 34(205):109–109, 1939.

[27] Milton Friedman. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics*, 11(1):86 – 92, 1940.

[28] Emmanuel Frécon and Mårten Stenius. Dive: a scaleable network architecture for distributed virtual environments. *Distributed Systems Engineering*, 5(3):91, sep 1998.

[29] Joshua S. Furtado, Hugh H. T. Liu, Gilbert Lai, Herve Lacheray, and Jason Desouza-Coelho. Comparative analysis of optitrack motion capture systems. In Farrokh Janabi-Sharifi and William Melek, editors, *Advances in Motion Sensing and Control for Robotic Applications*, pages 15–31, Cham, 2019. Springer International Publishing.

[30] Paul A. Games and John F. Howell. Pairwise multiple comparison procedures with unequal n's and/or variances: A monte carlo study. *Journal of Educational Statistics*, 1(2):113–125, 1976.

[31] Liang Gong, Henrik Söderlund, Leonard Bogojevic, Xiaoxia Chen, Anton Berce, Åsa Fast-Berglund, and Björn Johansson. Interaction design for multi-user virtual reality systems: An automotive case study. *Procedia CIRP*, 93:1259–1264, 2020. 53rd CIRP Conference on Manufacturing Systems 2020.

[32] The Khronos® OpenXR Working Group. "the openxr™ specification". `https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html`, 2023. Accessed: 2023-10-23.

[33] Robert Gruen, Eyal Ofek, Anthony Steed, Ran Gal, Mike Sinclair, and Mar Gonzalez-Franco. Measuring system visual latency through cognitive latency on video see-through ar devices. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 791–799, 2020.

[34] Ayah Hamad and Bochen Jia. How virtual reality technology has changed our lives: An overview of the current and potential applications and limitations. *International Journal of Environmental Research and Public Health*, 19:11278, 09 2022.

[35] Asim Hameed, Andrew Perkis, and Sebastian Möller. Evaluating hand-tracking interaction for performing motor-tasks in vr learning environments. In *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*, pages 219–224, 2021.

[36] Jan Hendrik Hammer and Jürgen Beyerer. Robust hand tracking in realtime using a single head-mounted rgb camera. In Masaaki Kurosu, editor, *Human-Computer Interaction. Interaction Modalities and Techniques*, pages 252–261, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[37] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D. Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, Asaf Nitzan, Gang Dong, Yuting Ye, Lingling Tao, Chengde Wan, and Robert Wang. Megatrack: Monochrome egocentric articulated hand-tracking for virtual reality. 39(4), aug 2020.

[38] Sebastian Herscher, Connor DeFanti, Nicholas Gregory Vitovitch, Corinne Brenner, Haijun Xia, Kris Layng, and Ken Perlin. Cavrn: An exploration and evaluation of a collective audience virtual reality nexus experience. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 1137–1150, New York, NY, USA, 2019. Association for Computing Machinery.

[39] Meta: Joel Hesch, Anna Kozminski, and Oskar Linde. "powered by ai: Oculus insight". `https://ai.meta.com/blog/powered-by-ai-oculus-insight/`, 2019. Accessed: 2023-10-09.

[40] Valentin Holzwarth, Joy Gisler, Christian Hirt, and Andreas Kunz. Comparing the accuracy and precision of steamvr tracking 2.0 and oculus quest 2 in a room scale setup. 03 2021.

[41] Lin Huang, Boshen Zhang, Zhilin Guo, Yang Xiao, Zhiguo Cao, and Junsong Yuan. Survey on depth and rgb image-based 3d hand shape and pose estimation. *Virtual Reality & Intelligent Hardware*, 3(3):207–234, 2021.

[42] The MathWorks Inc. "what is slam? 3 things you need to know". `https://www.mathworks.com/discovery/slam.html`, 2023. Accessed: 2023-10-09.

[43] Ultraleap Inc. "uh-003206-tc issue 6 leap motion controller data sheet". `https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf`, 2020. Accessed: 2023-10-06.

[44] Pascal Jansen, Fabian Fischbach, Jan Gugenheimer, Evgeny Stemasov, Julian Frommel, and Enrico Rukzio. Share: Enabling co-located asymmetric multi-user interaction for augmented reality head-mounted displays. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, page 459–471, New York, NY, USA, 2020. Association for Computing Machinery.

[45] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2020.

[46] Panagiotis T. Karfakis, Micael S. Couceiro, and David Portugal. Nr5g-sam: A slam framework for field robot applications based on 5g new radio. *Sensors*, 23(11), 2023.

[47] Gregory Kessler, Neff Walker, and Larry Hodges. Evaluation of the cyberglove(tm) as a whole hand input device. *ACM Transactions on Computer-Human Interaction*, 2, 12 1995.

[48] Chaowanan Khundam, Varunyu Vorachart, Patibut Preeyawongsakul, Witthaya Hosap, and Frédéric Noël. A comparative study of interaction time and usability of using controllers and hand tracking in virtual reality training. *Informatics*, 8(3), 2021.

[49] Chaowanan Khundam, Varunyu Vorachart, Patibut Preeyawongsakul, Witthaya Hosap, and Frédéric Noël. A comparative study of interaction time and usability of using controllers and hand tracking in virtual reality training. *Informatics*, 8:60, 09 2021.

[50] H. Kortier, Martin Schepers, Victor Sluiter, Peter Veltink, Alberto Leardini, and Rita Stagni. Ambulatory assesment of hand kinematics : using an instrumented glove. *Computer Standards & Interfaces - CSI*, 01 2012.

[51] William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[52] Bor-Woei Kuo, Hsun-Hao Chang, Yung-Chang Chen, and Shi-Yu Huang. A light-and-fast slam algorithm for robots in indoor environments using line segment map. *Hindawi Publishing Corporation Journal of Robotics*, 12, 01 2011.

[53] Pierre-Yves Lajoie, Benjamin Ramtoula, Fang Wu, and Giovanni Beltrame. Towards collaborative simultaneous localization and mapping: a survey of the current research landscape, 08 2021.

[54] Eike Langbehn, Gerd Bruder, and Frank Steinicke. Moving towards natural interaction between multiscale avatars in multi-user virtual environments. In *International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments 2015*, ICAT-EGVE 2015 : International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments, 2015.

[55] Eike Langbehn, Hannah Paulmann, Dennis Briddigkeit, Marc Barnes, Malte Husung, Kolja Kirsch, Daniel Neves Coelho, Tim Mayer, and Frank Steinicke. Frozen factory: A playful virtual experience for multiple co-located redirected walking users. In *SIGGRAPH Asia 2020 XR*, SA '20, New York, NY, USA, 2020. Association for Computing Machinery.

[56] Micheal Lanham. *Learn Unity ML-Agents Fundamentals of Unity Machine Learning: Incorporate new powerful ML algorithms such as Deep Reinforcement Learning for games*. Packt Publishing, 2018.

[57] Steven M. LaValle, Anna Yershova, Max Katsev, and Michael Antonov. Head tracking for the oculus rift. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 187–194, 2014.

[58] Kris Layng, Ken Perlin, Sebastian Herscher, Corinne Brenner, and Thomas Meduri. Cave: Making collective virtual narrative. In *ACM SIGGRAPH 2019 Art Gallery*, SIGGRAPH '19, New York, NY, USA, 2019. Association for Computing Machinery.

[59] Stefan Lee, Sven Bambach, David J. Crandall, John M. Franchak, and Chen Yu. This hand is my hand: A probabilistic approach to hand disambiguation in egocentric video. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 557–564, 2014.

[60] Howard Levene. *Robust Tests for Equality of Variance*, volume 2:, pages 278–292. Stanford University Press, 01 1960.

[61] Yue Li, Eugene Ch'ng, Shengdan Cai, and Simon See. Multiuser interaction with hybrid vr and ar for cultural heritage objects. In *2018 3rd Digital Heritage International Congress (DigitalHERITAGE) held jointly with 2018 24th International Conference on Virtual Systems & Multimedia (VSMM 2018)*, pages 1–8, 2018.

[62] Fanqing Lin, Connor Wilhelm, and Tony R. Martinez. Two-hand global 3d pose estimation using monocular RGB. *CoRR*, abs/2006.01320, 2020.

[63] Jiaojiao Lin, Fei Jiang, and Ruimin Shen. Hand-raising gesture detection in real classroom. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6453–6457, 2018.

[64] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee,

Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines. *CoRR*, abs/1906.08172, 2019.

[65] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A comprehensive survey of visual slam algorithms. *Robotics*, 11(1), 2022.

[66] Jameel Malik, Ahmed Elhayek, Fabrizio Nunnari, Kiran Varanasi, Kiarash Tamaddon, Alexis Héloir, and Didier Stricker. Deephps: End-to-end estimation of 3d hand pose and shape by learning from synthetic depth. *CoRR*, abs/1808.09208, 2018.

[67] Cristina Manresa-Yee, Javier Varona, Ramon Mas, and Francisco Perales. Hand tracking and gesture recognition for human-computer interaction. *Electronic Letters on Computer Vision and Image Analysis;*, ISSN 1577-5097 E:1, 01 2000.

[68] Alexander Masurovsky, Paul Chojecki, Detlef Runde, Mustafa Lafci, David Przewozny, and Michael Gaebler. Controller-free hand tracking for grab-and-place tasks in immersive virtual reality: Design elements and their empirical study. *Multimodal Technologies and Interaction*, 4(4), 2020.

[69] Fabrice Matulic, Taiga Kashima, Deniz Beker, Daichi Suzuo, Hiroshi Fujiwara, and Daniel Vogel. Above-screen fingertip tracking with a phone in virtual reality. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI EA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[70] Mark McGill, Jan Gugenheimer, and Euan Freeman. A quest for co-located mixed reality: Aligning and assessing slam tracking for same-space multi-user experiences. In *26th ACM Symposium on Virtual Reality Software and Technology*, VRST '20, New York, NY, USA, 2020. Association for Computing Machinery.

[71] Google MediaPipe. "hand landmarks detection guide". `https://developers.google.com/mediapipe/solutions/vision/hand_landmarker`, 2023. Accessed: 2023-12-15.

[72] Meta. "from the lab to the living room: The story behind facebook's oculus insight technology and a new era of consumer vr". `https://tech.facebook.com/reality-labs/2019/8/the-story-behind-oculus-insight-technology/`, 2019. Accessed: 2023-10-09.

[73] Meta. "set up hand tracking". `https://developer.oculus.com/documentation/unity/unity-handtracking/`, 2023. Accessed: 2023-10-06.

[74] Maximilian Metzner, Lorenz Krieg, Daniel Krüger, Tobias Ködel, and Jörg Franke. *Intuitive, VR- and Gesture-based Physical Interaction with Virtual Commissioning Simulation Models*, pages 11–20. 07 2020.

[75] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, 2351, 01 1994.

[76] C. Mizera, T. Delrieu, V. Weistroffer, C. Andriot, A. Decatoire, and J.-P. Gazeau. Evaluation of hand-tracking systems in teleoperation and virtual dexterous manipulation. *IEEE Sensors Journal*, 20(3):1642–1655, 2020.

[77] Curtiss Murphy. *Believable Dead Reckoning for Networked Games*, pages 307–328. 02 2011.

[78] Hyun Myung, Hae min Jeon, and Woo-Yeon Jeong. Virtual door algorithm for coverage path planning of mobile robot. In *2009 IEEE International Symposium on Industrial Electronics*, pages 658–663, 2009.

[79] Nadim Nachar. The mann-whitney u: A test for assessing whether two independent samples come from the same distribution. *Tutorials in Quantitative Methods for Psychology*, 4, 03 2008.

[80] Supreeth Narasimhaswamy, Thanh Nguyen, Mingzhen Huang, and Minh Hoai. Whose hands are these? hand detection and hand-body association in the wild. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4879–4889, 2022.

[81] D. Niehorster, L. Li, and M. Lappe. The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception*, 8, 2017.

[82] Brendan O'Flynn, Javier Torres, James Connolly, Joan Condell, Kevin Curran, and Philip Gardiner. Novel smart sensor glove for arthritis rehabiliation. In *2013 IEEE International Conference on Body Sensor Networks*, pages 1–6, 2013.

[83] Patrick Aggergaard Olin, Ahmad Mohammad Issa, Tiare Feuchtner, and Kaj Grønbæk. Designing for heterogeneous cross-device collaboration and social interaction in virtual reality. In *Proceedings of the 32nd Australian Conference on Human-Computer Interaction*, OzCHI '20, page 112–127, New York, NY, USA, 2021. Association for Computing Machinery.

[84] Open Source Computer Vision OpenCV. "detection of aruco markers". `https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html`, 2024. Accessed: 2024-01-11.

[85] Eva Ostertagova and Oskar Ostertag. Methodology and application of one-way anova. *American Journal of Mechanical Engineering*, 1:256–261, 11 2013.

[86] Kaitlyn M. Ouverson and Stephen B. Gilbert. A composite framework of co-located asymmetric virtual reality. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW1), apr 2021.

[87] Paschalis Panteleris, Iason Oikonomidis, and Antonis A. Argyros. Using a single RGB frame for real time 3d hand pose estimation in the wild. *CoRR*, abs/1712.03866, 2017.

[88] Daniel Passos and Bernhard Jung. *Measuring the Accuracy of Inside-Out Tracking in XR Devices Using a High-Precision Robotic Arm*, pages 19–26. 07 2020.

[89] Jérôme Perret and Emmanuel Vander Poorten. Touching virtual reality: a review of haptic gloves. 06 2018.

[90] Stephen Pheasant. *Bodyspace: Anthropometry, Ergonomics And The Design Of Work*. CRC Press, London, 2 edition, 2003.

[91] I. Podkosova, K. Vasylevska, C. Schoenauer, E. Vonach, P. Fikar, E. Bronederk, and H. Kaufmann. Immersivedeck: a large-scale wireless vr system for multiple users. In *2016 IEEE 9th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, pages 1–7, 2016.

[92] Iana Podkosova. *Walkable multi-user VR: the effects of physical and virtual colocation*. PhD thesis, Wien, 2018.

[93] N. Pretto and F. Poiesi. Towards gesture-based multi-user interactions in collaborative virtual environments. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W8:203–208, 2017.

[94] Sergi Pujades, Betty Mohler, Anne Thaler, Joachim Tesch, Naureen Mahmood, Nikolas Hesse, Heinrich H. Bülthoff, and Michael J. Black. The virtual caliper: Rapid creation of metrically accurate avatars from 3d measurements. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):1887–1897, 2019.

[95] Aylen Ricca, Amine Chellali, and Samir Otrnane. The influence of hand visualization in tool-based motor-skills training, a longitudinal study. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 103–112, 2021.

[96] Holger Salzmann, Jan Jacobs, and Bernd Froehlich. Collaborative Interaction in Co-Located Two-User Scenarios. In Michitaka Hirose, Dieter Schmalstieg, Chadwick A. Wingrave, and Kunihiro Nishimura, editors, *Joint Virtual Reality Conference of EGVE - ICAT - EuroVR*. The Eurographics Association, 2009.

[97] Muhamad Risqi U. Saputra, Andrew Markham, and Niki Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Comput. Surv.*, 51(2), feb 2018.

[98] Daniel Schneider, Verena Biener, Alexander Otte, Travis Gesslein, Philipp Gagel, Cuauhtli Campos, Klen Copic Pucihar, Matjaz Kljun, Eyal Ofek, Michel Pahud, Per Ola Kristensson, and Jens Grubert. Accuracy evaluation of touch tasks in commodity virtual and augmented reality head-mounted displays. *CoRR*, abs/2109.10607, 2021.

[99] Daniel Schneider, Alexander Otte, Axel Simon Kublin, Alexander Martschenko, Per Ola Kristensson, Eyal Ofek, Michel Pahud, and Jens Grubert. Accuracy of commodity finger tracking systems for virtual reality head-mounted displays. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 804–805, 2020.

[100] H. Schupp. *Elementargeometrie*. Number Bd. 1 in Grundkurs Mathematik. Schöningh, 1977.

[101] Alexander Schäfer, Gerd Reis, and Didier Stricker. Comparing controller with the hand gestures pinch and grab for picking up and placing virtual objects, 2022.

[102] S. S. SHAPIRO and M. B. WILK. An analysis of variance test for normality (complete samples)†. *Biometrika*, 52(3-4):591–611, 12 1965.

[103] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 3633–3642, New York, NY, USA, 2015. Association for Computing Machinery.

[104] Yangming Shi, Jing Du, Sarel Lavy, and Dong Zhao. A multiuser shared virtual environment for facility management. *Procedia Engineering*, 145:120–127, 2016. ICSDEC 2016 – Integrating Data Science, Construction and Sustainability.

[105] TIGA: Suzi Stephenson. "tiga survey reveals that unity 3d engine dominates the uk third party engine market". `https://tiga.org/news/tiga-survey-reveals-that-unity-3d-engine-dominates-the-uk-third-party-engine-market`, 2019. Accessed: 2023-10-06.

[106] Stephan Streuber and Astros Chatziastros. Human interaction in multi-user virtual reality. *Proceedings of the 10th International Conference on Humans and Computers (HC 2007), 1-7 (2007)*, 01 2007.

[107] Z. Sun, Y. Hu, and X. Shen. Two-hand pose estimation from the non-cropped rgb image with self-attention based network. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 248–255, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

[108] Philipp Sykownik, Sukran Karaosmanoglu, Katharina Emmerich, Frank Steinicke, and Maic Masuch. Vr almost there: Simulating co-located multiplayer experiences in social virtual reality. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[109] Unity Technologies. "creating custom packages". `https://docs.unity3d.com/Manual/CustomPackages.html`, 2023. Accessed: 2023-10-06.

[110] Pablo Temoche, Esmitt Ramirez, and Omaira Rodríguez. A low-cost data glove for virtual reality. pages TCG 31–36, 05 2012.

[111] Silvia Terrile, Jesus Miguelañez, and Antonio Barrientos. A soft haptic glove actuated with shape memory alloy and flexible stretch sensors. *Sensors*, 21(16), 2021.

[112] Shantanu Tilak, Michael Glassman, Irina Kuznetcova, Joshua Peri, Qiannan Wang, Ziye Wen, and Amanda Walling. Multi-user virtual environments (muves) as alternative lifeworlds: Transformative learning in cyberspace. *Journal of Transformative Education*, 18(4):310–337, 2020.

[113] Satoshi Tsutsui, Yanwei Fu, and David Crandall. Whose hand is this? person identification from egocentric hand gestures, 2020.

[114] Kyriaki A. Tychola, Ioannis Tsimperidis, and George A. Papakostas. On 3d reconstruction using rgb-d cameras. *Digital*, 2(3):401–421, 2022.

[115] Toin Villar. "what is the metaverse?". `https://www.makeuseof.com/what-is-the-metaverse/`, 2019. Accessed: 2023-10-10.

[116] Jan-Niklas Voigt-Antons, Tanja Kojic, Danish Ali, and Sebastian Möller. Influence of hand tracking as a way of interaction in virtual reality on user experience. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–4, 2020.

[117] Aleš Vysocký, Stefan Grushko, Petr Oščádal, Tomáš Kot, Ján Babjak, Rudolf Jánoš, Marek Sukop, and Zdenko Bobovský. Analysis of precision and stability of hand tracking with leap motion sensor. *Sensors*, 20(15), 2020.

[118] David Waller, E.R. Bachmann, Eric Hodgson, and Andrew Beall. The hive: A huge immersive virtual environment for research in spatial cognition. *Behavior research methods*, 39:835–43, 12 2007.

[119] Jiayi Wang, Franziska Mueller, Florian Bernard, Suzanne Sorli, Oleksandr Sotnychenko, Neng Qian, Miguel A. Otaduy, Dan Casas, and Christian Theobalt. Rgb2hands: Real-time tracking of 3d hand interactions from monocular rgb video. 39(6), nov 2020.

[120] Zheng Wang, Luca Mastrogiacomo, Fiorenzo Franceschini, and Paul Maropoulos. Experimental comparison of dynamic tracking performance of igps and laser tracker. *The International Journal of Advanced Manufacturing Technology*, 56, September 2011.

[121] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors (Basel, Switzerland)*, 13:6380–6393, 05 2013.

[122] T. Weissker, P. Tornow, and B. Froehlich. Tracking multiple collocated htc vive setups in a common coordinate system. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 592–593, 2020.

[123] B. L. Welch. The generalisation of student's problems when several different population variances are involved. *Biometrika*, 34(1-2):28–35, 01 1947.

[124] Brian Williams, Georg Klein, and Ian Reid. Real-time slam relocalisation. pages 1–8, 01 2007.

[125] Sen-Zhe Xu, Jia-Hong Liu, Miao Wang, Fang-Lue Zhang, and Song-Hai Zhang. Multi-user redirected walking in separate physical spaces for online vr scenarios, 2022.

[126] Yu Xu and Xi'an Zhu. The research and application of data glove in virtual interaction system. *Advanced Materials Research*, 989-994:2057–2061, 07 2014.

[127] Umema Zafar, Shafiq-Ur-Rahman, Naila Hamid, Junaid Ahsan, and Nimra Zafar. Correlation between height and hand size, and predicting height on the basis of age, gender and hand size. *Journal of Medical Sciences (Peshawar)*, 25:425–428, 10 2017.

[128] Faisal Zaman. [dc] improving multi-user interaction for mixed reality telecollaboration. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 940–941, 2022.

[129] José Zariffa and Milos Popovic. Hand contour detection in wearable camera video using an adaptive histogram region of interest. *Journal of neuroengineering and rehabilitation*, 10:114, 12 2013.

[130] Fangfang Zhang, Valentin Bazarevsky, Andrey Vakunov, A. Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *ArXiv*, abs/2006.10214, 2020.

[131] Jingbo Zhao, Ruize An, Ruolin Xu, and Banghao Lin. Comparing hand gestures and a gamepad interface for locomotion in virtual environments. *International Journal of Human-Computer Studies*, 166:102868, 2022.

[132] Huayi Zhou, Fei Jiang, and Ruimin Shen. Who are raising their hands? hand-raiser seeking based on object detection and pose estimation. In Jun Zhu and Ichiro Takeuchi, editors, *Proceedings of The 10th Asian Conference on Machine Learning*, volume 95 of *Proceedings of Machine Learning Research*, pages 470–485. PMLR, 14–16 Nov 2018.

[133] Thomas G. Zimmerman, Jaron Lanier, Chuck Blanchard, Steve Bryson, and Young Harvill. A hand gesture interface device. *SIGCHI Bull.*, 17(SI):189–192, may 1986.

[134] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. https://arxiv.org/abs/1705.01389.

List of Figures

List of Tables