Improving User Performance in Rehabilitation Exercises

by

Renz Jethro Roque Ocampo

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

BIOMEDICAL ENGINEERING

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Disabling events such as stroke affect millions of people worldwide, causing a need for efficient and functional rehabilitation therapies in order for patients to regain motor function for reintegration back into their normal lives. Rehabilitation regimes often involve performing exercises that mimic the movements done in activities of daily living. These are sometimes complemented with serious games controlled through a robotic user interface to increase the motivation of the patients, further increasing the likelihood of success of the therapy. However, alongside physical disability, some patients (e.g., stroke patients) develop cognitive deficiencies that affect their ability to think, plan, and carry out tasks. In such cases, serious games, which are commonly displayed on a 2D monitor to the patient, may be too hard for patients due to the spatial disconnect between the visual coordinate frame (screen frame) and the hand coordinate frame (robotic user interface frame). Patients will have to do mental transformations to align their hand movements with their movements on-screen. The colocation of visual and motor frames for rehabilitation in commercial devices is still in its infancy, and while there are research regarding the use of visual-motor colocation in rehabilitation, its effectiveness has not yet been explored.

This thesis presents a study of the effectiveness of visual-motor colocation in rehabilitation exercises by integrating augmented reality in serious games to achieve the above-mentioned colocation. A technique called projection mapping is utilized to project digitally constructed objects onto the real-world environment. Physical interaction with these objects is handled through a haptic user interface. The system is comprised of a projector, a game engine to create the virtual environment, a haptic user interface to interact and receive force feedback on the virtual objects, and a depth sensor to implement head tracking in 3D scenarios.

The system design and the investigations in this study consist of two stages: first implementing visual-haptic colocation in a 2D spatial augmented-reality display in Chapter 3, and then further extending the work into a 3D environment in Chapter 4. The 2D case involves a task where reaching motions are performed using a 2D planar haptic robot. For the 3D case, three tasks are presented, each requiring a combination of spatial accuracy, awareness, and manipulation. Disability-induced cognitive deficiency is simulated on able-bodied participants by putting them under a cognitive load while performing the tasks. Each of the tasks in the 2D and 3D cases are compared to their non-colocated counterpart (tasks displayed on a 2D screen placed in front of the user) in terms of several user performance indicators. Results show a significant increase in user performance when visual and motor frames are colocated for both 2D and 3D cases. Furthermore, one of the tasks in 3D showed that visual-motor colocation can alleviate the negative effects of cognitive loading.

Finally, after the validation of the effectiveness of AR in robot-assisted therapy, we combine AR with a heavy-duty robot in Chapter 5 and explore the use of this robot-AR system in occupational rehabilitation and functional capacity evaluations. The biomechanics of the user's arm while performing the task with the robot-AR system is compared with their arm biomechanics for an equivalent real-world task. An analysis for similarity of the arm biomechanics is carried out to determine if using the robot-AR system can produce the same upper-limb movements as in conventional rehabilitation practices.

By increasing the user performance, which consequently increases the likelihood of success in performing the exercise, this work of bridging the spatial disparity between two frames can potentially improve the efficiency of current rehabilitation practices that use serious games for therapy. It has been shown that users who are set up to succeed more are more likely engaged in rehabilitation. It becomes a positive feedback loop where as the patient's performance improves with practice, this improvement allows the patient to do even more. While not all patients achieve this positive feedback structure, we hope to make it easier for patients to reach this "threshold."

Preface

This thesis is an original work by Renz Ocampo. The research project, of which this thesis is a part, received research ethics approval from the University of Alberta Research Ethics Board, Project Name ”Measuring the mechanical impedance of the upper limb using a rehabilitation robot”, No. MS7_Pro00033955. April 27, 2017. Initially used as the ethics approval by Matthew Dyck in 2013, this ethics approval was revised to encompass the work done in this thesis.

Chapters 3 and 4 of this thesis are my original work. Chapter 3 has been published as Renz Ocampo and Mahdi Tavakoli, “Visual-Haptic Colocation in Robotic Rehabilitation Exercises Using a 2D Augmented-Reality Display,” The International Symposium on Medical Robotics (ISMR), Atlanta, GA, 2019. Chapter 4 has been published as Renz Ocampo and Mahdi Tavakoli, “ Improving User Performance in Haptics-Based Rehabilitation Exercises by Colocation of User’s Visual and Motor Axes via a 3D Augmented-Reality Display,” IEEE Robotics and Automation Letters, 2019. In press. For both works, I was responsible for the development of the system, data collection, analysis, and manuscript composition. Mahdi Tavakoli was the supervisory author and was involved with concept formation and manuscript composition.

Chapter 5 of this thesis is currently in preparation for submission and is a collaborative work with Jason Fong. I was responsible for implementing augmented-reality and programming the virtual environment. Jason Fong handled the admittance control for the serial manipulator and the majority of the analysis. The manuscript composition, data collection, and experimentation were done by both of us in equal parts.

Dedication

To my parents, Roland and Jasmin:
Thank you for the love, care, and support you have given me throughout this journey and for always showing concern to keep myself from getting off track. Thank you also for ensuring I never go hungry.

To my brother, Riel:
Thank you for helping me stay relaxed and stress-free. The distractions are always welcome.

To God:
Thank you for giving me the perseverance to move forward, for helping me stay composed when experiments are not working, and for granting me this new accomplishment in life.

Acknowledgements

I would like to express my most sincere gratitude and appreciation for my supervisor, Dr. Mahdi Tavakoli, who has guided and encouraged me throughout the entirety of my degree. He has always put in the effort to make himself available to help out and give advice to his students. Thank you for being patient with me and mentoring me every step of the way.

Thank you, Dr. Mahdi Tavakoli, Dr. Edmond Lou, and Dr. Hossein Rouhani for serving on the examination committee and Dr. Kambiz Moez as the chair.

Thank you to my family for their love and support.

Thank you also to my friends and fellow lab members in the Biorobotic and Telerobotic Systems Group. The insight and experiences you have shared with me have truly been of great help and inspiration. Especially, I would like to thank Jason Fong for the collaborative work we have done for the final chapter which would not have been possible without his expertise.

Funding for this research has been provided in part by the Natural Sciences and Engineering Research Council (NSERC), the Canada Foundation for Innovation (CFI), the Alberta Innovation and Advanced Education Ministry, and Quanser Inc.

Contents

List of Tables

List of Figures

List of Acronyms

$2D$	2 Dimensions
$3D$	3 Dimensions
$ADLs$	Activities of Daily Living
$ANOVA$	Analysis of Variance
AR	Augmented Reality
CL	Cognitive Loading
DOF	Degree of Freedom
$E2S$	Elbow-to-Shoulder
FCE	Functional Capacity Evaluation
FDR	False Discovery Rate
$H2E$	Hand-to-Elbow
HD^2	High Definition Haptic Device
HMD	Head Mounted Display
KS	Kolmogorov–Smirnov
NN	Nearest Neighbor
OST	Optical See-Through
$PHRI$	Physical Human-Robot Interaction
$PILE$	Progressive Isoinertial Lifting Evaluation
RM	Repeated Measures
SD	Standard Deviation
VST	Video See-Through
VR	Virtual Reality

Chapter 1

Introduction

1.1 Motivation

The demand for motor rehabilitation has soared to new heights in the last few decades due to the increase in disabling events such as stroke, injury, and accidents. Using stroke as an example, approximately 62,000 cases are reported in Canada each year, making it the 3rd leading cause of death in Canada. With 405,000 Canadians living with the effects of stroke, the cost to the Canadian economy totals more than $20.9 billion per year [1]. As such, the need for efficient forms of therapy is on the rise in order to reduce the human and capital resources required and to better assist patients in regaining their lost functions.

The ability to do activities of daily living (ADL) such as eating, dressing, and self-care is typically negatively affected by a disabling event. Therefore, the fundamental movements that make up ADLs such as point-to-point reaching are commonly used as exercises for rehabilitation. Traditionally, these are done using hand-over-hand therapy in which the therapist directs the patient's movement. To be effective in regaining motor function, however, multiple repetitions of the task is needed, requiring the therapist to commit long hours per patient and therefore only allows a few patients to receive therapy. With lengthy therapy sessions required by patients, each with labour-intensive activities, a heavy burden is placed on the health care system. Thus, the inclusion of robots in the rehabilitation environment is desirable. Robots can reproduce tasks with unfailing accuracy for multiple repetitions without feeling the effects of fatigue. This assistance to the therapist allows more patients to be handled in the clinic, thereby permitting for more efficient rehabilitation.

Low patient motivation, however, remains to be an issue in therapy even with the addition of robotics. As robots allow for a reduced therapist intervention, the patients themselves lose motivation due to the lack of encouragement, entertainment, and human interaction [2][3][4]. Motivation is seen as an important predictor in successful rehabilitation outcomes. Serious games, which are video games designed for purposes other than pure entertainment, have been shown to increase patient motivation [5][6]. By combining serious games and robot-assisted rehabilitation, the patient becomes engaged in the exercise and may even "forget" that they are in a rehabilitation training session due to the immersion [7]. It has been shown that the combination of the two technologies in rehabilitation training leads to better outcomes than robotics assistance alone [8][9].

The robotic interfaces can also provide accurate measurements of movement and forces that can be used to give a quantitative analysis of performance, a factor that is lacking in a therapist's subjective observation. Giving quantitative feedback to inform the patients about small improvements in their success scores or limb movements further encourages them to continue on with their therapy.

While serious games aid in motivating patients during rehabilitation exercises, they lack the sense of realism found in traditional rehabilitation practices. Real-world tasks involve interaction with real objects such as peg-in-the-hole insertion, block stacking, and pick-and-place operations. Patients touch and see the objects they are interacting with at the exact same location since both the visual and haptic frames are aligned. However, the games used for rehabilitation are typically shown on a 2D screen in front of the user. The disconnect between the patient's arm movement axis and how they see their cursor or avatar move on the screen may unnecessarily impose a mental burden on the patients to match their limb movements with what they visually see on the display. Furthermore, the scaling of movements might need to be accounted for if the workspace and screen are of different sizes. For patients whose cognitive functions are negatively affected by events such a stroke or injury and who require upper-limb neurorehabilitation, the spatial disparity might make it more difficult to perform the task compared to those with no cognitive deficiency.

1.1.1 Objective

The principal idea of this thesis is that by colocating the visual and motor frames of the workspace for the user, the user's performance in rehabilitation exercises will be increased due to the reduced mental load. Our aim is to have enhanced patient engagement and thus better rehabilitation outcomes. However, performing an actual investigation requires a longitudinal patient study which is beyond the scope of this thesis.

1.1.2 Organization of the Thesis

Chapter 2 presents the work done in the literature that provides information to help understand the context of this thesis. A brief overview of robotic rehabilitation is discussed along with serious games used in rehabilitation. This is followed by a more comprehensive description of the different types of visual display techniques used in rehabilitation.

Chapter 3 investigates the effectiveness of visual-haptic colocation in a 2D rehabilitative task. Spatial AR (projection mapping) is utilized as the display method to align the visual and motor frames. The task is a serious game that involves doing reaching motions around the workspace using a 2 Degree-Of-Freedom (DOF) planar rehabilitation robot. The "patients" are simulated by able-bodied participants that simultaneously perform a counting down arithmetic operation while doing the task. Three independent parameters are switched on or off in the experiments: visual-motor colocation, haptic feedback, and cognitive loading. User task performance is measured based on time to completion for the different parameter combinations.

Chapter 4 builds upon the findings of Chapter 3 and extends the work to a 3D spatial AR system. This chapter uses a 6-DOF haptic user interface to interact with the projected digital objects. Active 3D shutter glasses are worn by the participants to provide depth information to the user. Three tasks are presented to each participant: Snapping, Catching, and Ball Dropping. Since the tasks differ in both objectives and their interaction with the virtual environment, haptic feedback is not equally represented in the tasks and is therefore always turned on when applicable. Only two parameters are manipulated for each task: visual-motor colocation and cognitive loading. User task performance is measured based on the success scores earned per task.

Chapter 5 applies the AR concept to another area of rehabilitation, occupational (or vocational) rehabilitation of injured workers. A heavy-duty 7-DOF serial manipulator is combined with AR to recreate the physical dynamics of functional tasks to help regain an injured worker's functional capabilities and allow for a return to employment. In this chapter, we simulate a painting task that focuses on training up-down hand movements by having the user paint a virtual wall. The system's performance and efficacy are validated by comparing the user's arm biomechanics when using the robot-AR setup versus their arm biomechanics during the performance of the physical version of the same task.

Chapter 6 provides a summary of the research findings and possible areas of future work.

1.1.3 Patient Simulation

It is important to note that the patients in this thesis are *simulated*. Effects of cognitive deficiency, such as the inability to give full attention to the task, is simulated by cognitively loading able-bodied participants using an arithmetic task. While this does not fully mimic the effects of mental disability, the main consideration is the extent participants are burdened by the arithmetic operation such that they produce performance results that an actual patient with a disability might have.

1.1.4 Contribution of the Thesis

This thesis makes contributions in the investigation of the effectiveness of visual-motor colocation in rehabilitation exercises for both 2D and 3D visual displays in Chapters 3 and 4. Furthermore, the combination of a robotic user interface coupled with haptics, AR, and cognitive loading applied to serious games for rehabilitation has not yet been examined. The focus is on the comparison of user performance between when visual-motor colocation is toggled on and off. This performance is also examined against tasks without cognitive loading to analyze how much visual-motor colocation can mitigate the negative cognitive effects of a disability.

In Chapter 5, current practices for occupational rehabilitation involve real physical objects across multiple setups that together occupy a large area of an occupational rehabilitation facility. Other systems that combine these exercises into one adjustable machine lack the visual representation of the tasks patients are performing. Our robot-AR system aims to solve these issues in one unified system.

Chapter 2

Literature Review

This chapter presents the background information required to understand the fundamental concepts found in the thesis. An overview of the surrounding literature is discussed in the following sections. Section 2.1 provides a brief history of robotics being used for motor rehabilitation. Section 2.2 introduces the role of serious games in the field of rehabilitation. The visual display methods, namely virtual reality (VR) and augmented reality (AR), used for serious games in robot-assisted rehabilitation are discussed in Section 2.3.

2.1 Robot-Assisted Rehabilitation

Dedicated to assisting and augmenting motor function rehabilitation using robotic devices, research in rehabilitation robotics began as a way to find a solution that alleviates therapists' stress and produces more efficient rehabilitation techniques. Since traditional upper-limb rehabilitation therapy is administered through a hand-over-hand manner by the therapists, physical fatigue and pressure is built up through the long hours of commitment required per patient. Spurred by a robot's ability to produce high-intensity, repetitive, and precise motions, researchers have taken interest in rehabilitation applications for robotics.

Initially, most robots used in rehabilitation were for assistive purposes. These robots did not aim to help to regain the lost motor function of the patient, but rather they aimed to assist the patient in performing activities of daily living. These are commonly seen as robots attached to wheelchairs to assist in eating and drinking, grabbing objects, and mobility [10]. It was not until the late 1980s when researchers started to pursue rehabilitation robotics for actual therapy use. In 1988, two double-link planar robots were coupled with a patient's lower limb to provide continuous passive motion for rehabilitation [11]. This was soon followed by an upper-limb rehabilitation device in 1992, the MIT-MANUS, which was used for planar shoulder-and-elbow therapy [12]. Upper-limb rehabilitative devices were further developed after the advent of the MIT-MANUS. These include devices such as the Mirror-Image Movement Enabler (MIME) robotic device, which improved muscle movements through mirror-image training [13], and the Assisted Rehabilitation and Measurement (ARM) Guide, which functions both as an assessment and rehabilitative tool [14]. Robotic rehabilitation that targeted other areas of the body surfaced in the 2000s. These robotic devices allowed rehabilitation for areas such as the wrist [15], hand, and finger [16] for the upper-limb, and gait and ankle training [17][18] for the lower limb.

While the idea of robotic therapy is alluring, researchers sought to investigate its effects in regaining motor function in rehabilitation. Improvement of motor function have been shown in studies performed with the MIT-MANUS. Post-stroke patients were recruited for the MIT-MANUS study and results have shown a statistically significant reduction in shoulder and elbow impairment compared to the group that underwent conventional therapy [19]. Likewise, with the MIME system, post-stroke patients trained with the MIME over an eight-week period showed an improvement in reach and strength of the affected limb [20]. Based on the Fugl-Meyer score, the MIME group produced better results than the conventional therapy group in the two months of therapy. These studies show the potential of robotic therapy in the rehabilitation practice.

The introduction of robotic devices for rehabilitation have opened up new possibilities in improving the efficacy of traditional rehabilitation techniques. The results of robotic therapy show promise; however, the repetitive nature of these exercises can make it a uninteresting task for the patient, thereby decreasing their motivation. Studies have shown that incorporating games alongside rehabilitation practices increases the patient's motivation during therapy [3][21][5].

2.2 Serious Games

First coined by Abt in his 1970 book [22] and popularized by Sawyer in 2002 [23], serious games have become a widely researched field of study garnering a worldwide market worth of €1.5 billion in 2010 [24]. The exact origin of using games for a purpose other than entertainment is unknown; however, historical data shows evidence of its concept tracing as far back as ancient Greece. Plato acknowledged that for children, play can have an effect on their development into adulthood [25]. The early 1900s showed interest in games as a supplement in the field of education. In 1902, *The Landlord's Game* was designed to be a "practical demonstration of the present system of land-grabbing with all its usual outcomes and consequences" [26]. It was then used by Scott Nearing, a professor at the University of Pennsylvania, who used it for teaching [27]. In the next few decades, it became the game of *Monopoly* that we have all come to know and enjoy. Since the video game industry had not yet taken form at the time, paper-based educational games in the 1960s briefly gained attention but never rose in popularity [28][22]. By the 1970s, computer games such as *The Oregon Trail* [29], which taught children about 19th-century pioneer life, started to appear although these were mainly games for educational use. It was not until the early 2000s when fueled by the advancements in hardware, game development, and its success in the commercial market, that researchers worldwide took an interest in novel areas of research that video games can be applied to. Defined as games that serve a main purpose other than pure entertainment, serious games have been expanding in areas such as politics [30], military [31], sports training [32], and health [33]. Serious games can be presented through any form of technology and can be in any genre. For instance, even commercial platforms intended for entertainment, the Nintendo Wii, the PlayStation 2, and the Microsoft Kinect, have been repurposed for research in the rehabilitation field [34][35][36]. Various conferences and seminars have emerged in response to the growing interest. In 2010, the Serious Play Conference [37] began a leadership conference for developers of serious games/simulations coming from different fields of expertise from both industry and academia. To promote interdisciplinary research within the field, IEEE launched the first serious games conference in 2009, dubbed as VS-GAMES: Games and Virtual Worlds for Serious Applications. Limitless in potential, games allow for adding entertainment in approaches to education, training, or rehabilitation. In essence, the purpose is to create an enjoyable environment for otherwise tedious activities.

2.2.1 Serious Games in Rehabilitation

Patient Motivation

Patient motivation is regarded as an important factor in predicting success rates for rehabilitation [38]. The physical and emotional impacts induced by events such as stroke affect the willingness of the patient to participate. While this motivation to rehabilitate can be influenced by multiple factors in the rehabilitation environment such as family members, staff, and other stroke patients, a significant element lies in rehabilitation exercises itself. Some patients have a lack of information and understanding of the nature of their rehabilitation exercises [39]. Coupled with the long sessions and repetitive motions that are involved [40], patients become uncertain about their rehabilitation outcomes. As a consequence, poor patient participation causes patients to have longer inpatient rehabilitation stay and poorer motor improvement results [41].

Motivation through games

The addition of serious games in the rehabilitation environment to augment physical therapeutic exercises has been shown to produce positive outcomes. Since the games can promote the use of both physical motions and mental processes, the patient becomes actively focused while doing exercises. One of the earlier documented applications of computer games in rehabilitation research was done in 1993 for promoting arm reach using a *Simon* game in which patients are instructed to repeat the sequence of flashing lights by pressing coloured buttons [42]. By adding the game, it structures the exercise such that "performance components are not acquired through random activity or mindless exercise; rather they are acquired through active, goal-directed interaction with the environment" [43].

The availability of various interfaces for serious gaming allows for adaptability and variety for the patient. From keyboards and mice, to body tracking and head-mounted displays, a multitude of devices can be integrated in games. Robots and haptic interfaces can provide haptic feedback to the patients to enable interaction with digital objects. For example, technologies like The Java Therapy System [44] allow for different interfaces (traditional mice, force feedback mice, force feedback joystick) to be used with games.

Serious gaming for rehabilitation does not end outside the doors of the therapy clinics; it plays a key role in home-based rehabilitation. Patients who are discharged from the hospitals and are sent home, but are required to do exercises by themselves, lack the motivational support from staff and peers in the clinics. This also applies to patients without access to the clinics due to distance or other circumstances. Self-exercise often leads to low motivation and less adherence to the required daily workout dose when lacking motivational support. Engaging patients at home with serious games for rehabilitation promotes adherence to the therapy.

Sources of motivation also come from the environment surrounding the patient. The therapist, the patient's family, and peers in the rehabilitation clinics are all factors that can affect the patient's motivation. Including these elements in a serious game further motivates patients by making the exercise seem more comparable to a social activity. Jadhav et al. [45] developed a system that allowed the therapist to perform an active role in adjusting the complexity of the training regimen while the patient does the exercise from the remote environment. Multiplayer games with peers offer the patients a shared experience and a sense of social acceptance [46]. Being able to cooperate or compete with one another heightens the enjoyment and can potentially produce a more intensive exercise than playing alone [47]. Furthermore, differences in performance levels for an able-bodied person and a post-stroke patient could be equalized such that they can both take part in a rehabilitative game [48].

There is significance in the type of information the patient receives during their exercise. In traditional rehabilitation practices, this may be in the form of distinct changes in the difficulty of an exercise, location of the user's hand with respect to the target position, or general commentary feedback from the therapist. Robotic therapy takes advantage of the ability to record quantitative information to accurately measure even minor improvements in the patient's actions [49]. Coupled with the games, the recorded data could be transformed into an exciting challenge to beat such as using the patient's progress as high scores or achievements to work towards to further engage the patients [33][50].

Moreover, since the confidence of patients may decrease when recognizing an increased intensity or difficulty in the task, the task difficulty could also be discreetly changed unbeknown to the patient. The game serves to aid in taking away attention from the change, thereby producing a masking effect.

Using the games as a distractor can also be used to alleviate pain. Patients can feel discouraged to continue their rehabilitation exercises when knowing that doing so inflicts pain. The same case is often seen in pediatrics when children become difficult to manage due to the fear of needles. A study on burn patients undergoing burn rehabilitation therapy reported that less pain was experienced when the patients were distracted by VR. The patients spent less time thinking about the pain while distracted

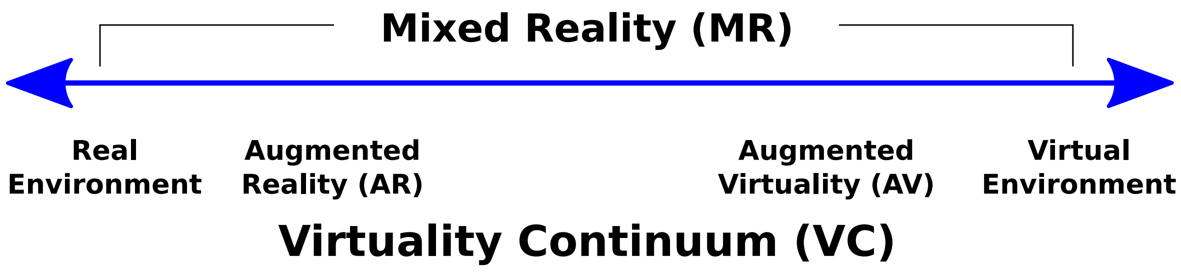


Figure 2.1: The Virtuality Continuum introduced by Milgram et al. to categorize different mixed reality environments.

[51]. The pain can also be reattributed as a different sensation (e.g., a needle's sting can be represented as a warm object on the arm) in the VR environment to further draw attention away from it [52][53].

In summary, serious games provide entertainment to users while having a primary objective of developing skills. In the rehabilitation case, the games encourage patients to keep up with their training regimen in several possible ways. Having a variety of interfaces available for use can make the therapy adaptable to different kinds of patients and open up avenues to various styles of gaming. The games can be a source of motivation for people doing home-based rehabilitation or a way to socially connect with others through a multiplayer exercise. They can be used to present patient performance feedback in engaging ways such as challenges to beat, milestones, and high scores. Alternatively, they can also aid in masking information such as changes in difficulty levels or by distracting the patients to keep their attention away from the physical pain during exercises.

2.3 Virtual Reality and Augmented Reality

Game environments can be shown to users through different display techniques. This can range from a typical 2D computer screen to more state-of-the-art technologies such as the Microsoft Hololens. The environment presented to the user can be described through the Virtuality Continuum (VC) [54].

The concept of the VC illustrates how the presentation of objects is not limited to a purely virtual or purely real environment. Looking through a cellphone camera and seeing the real world is considered as part of the left end of the continuum, the real environment. On the other hand, games like *Super Mario Brothers* belong purely to the virtual environment (or virtual reality) on the right end of the continuum. The term “Mixed Reality” arises when these two environments are combined within a single display. Depending on the proportion of real and virtual environments represented, the result can be categorized as either Augmented Reality (AR) or Augmented Virtuality (AV). In AR, virtual objects are integrated into the real environment. It involves the augmentation of the real world with virtual (computer generated) objects. At one’s viewpoint, these objects are seamlessly integrated as if they physically exist alongside the real-world objects. This potentially allows users to digitally interact with their surrounding environment. Games such as *Pokemon Go*, where virtual objects are overlaid on the camera feed, are considered as AR. Conversely, AV adds real objects into the virtual environment. This can be imagined as displaying the user’s actual hand in a virtual environment to interact with virtual objects. This is not to be confused with the representation of AR where the entire real environment is overlaid with a virtual one in which only certain real objects are unmasked such that they appear in the virtualized world [55]. It is important to note that these environments are not restricted by a specific type of display method such as a head-mounted display. Even a simple 2D screen would be capable of displaying these environments.

Although the majority of AR and VR development is geared for visual use, there is some research that applies to other senses also [56]. For example, applications for AR in surgery also involve active sensorimotor augmentation techniques to either provide supplementary perceptual information or to augment motor control capabilities of the surgeon during robotic-assisted surgeries [57].

In this thesis, AR is added to the robotic system through the use of visual displays to enhance the user’s perception of the tasks. These can come in many forms when it comes to presenting VR or AR. These displays include, but are not limited to computer screens, projection surfaces, HMDs, or semi-transparent mirrors. The following subsection presents a brief overview of the display techniques commonly used for VR and AR as well as the recently developed systems that can be found in rehabilitation research literature. These examples will be further expanded upon in Chapters 3 and 4.

2.3.1 Virtual Reality Displays

There are two main ways that VR is presented: non-immersive VR through a 2D computer screen, or immersive 3D VR using an HMD. The ReJoyce by Rehabtronics [58] is an example of a non-immersive VR system controlled by a passive robotic interface. A multitude of games can be selected and played with using the various components that are incorporated the robotic interface. As for immersive VR, the user is brought into a completely virtual 3D environment. The virtual surroundings can be configured in any manner regardless of genre or theme, allowing for creative ways of incorporating it into different applications in the healthcare field. Immersive VR has been used in studies such as treating phobia [59], assessing mild traumatic brain injury [60], gait therapy [61], hand rehabilitation [62], and so on.

2.3.2 Augmented Reality Displays

Video see-through (VST), optical see-through (OST), and spatial AR (i.e. projection-based AR) are the three most common ways of displaying AR [56]. VST takes a video feed of the real world and then virtual objects are directly overlaid on the display on the live video [63]. Registering the virtual objects can be done using fiducial markers attached to real-world surfaces [55]. By digitizing the reality through a camera, it is much easier to manipulate the environment using image processing tools as both virtual objects and the real world are now in the same digital space. This means aspects such as contrast, orientation, and size of the virtual object can be calibrated more easily with the real world. With OST, the virtual and real aspects interact through a semi-transparent mirror [64]. Essentially, the user can see through the display, and the virtual objects are reflected on the display for the user to see. OSTs let the user experience their environment directly without relying on the screen of a VST and therefore the resolution quality is as good as the user's eyesight. However, it may be bulkier as cameras, monitors, and a mirror are needed to assess the environment and display virtual objects. Finally, projection is another method to implement AR [65]. Projection solves the problem of requiring HMDs or looking at a separate screen to see the computer-generated images. It can be used on both flat and 3D surfaces with cameras or motion trackers for direct interaction. Techniques such as projection mapping are available for the virtual objects to "pop-out" and seem one with the environment. However, projection is limited to low light working environments since it can be easily overpowered by other light sources.

Chapter 3

Investigation of User Performance Improvement in a 2D Visual-Haptic Colocated Rehabilitation Task

3.1 Introduction

The demand for rehabilitation services has grown significantly with the aging of population. Patients who have suffered disabling events such as stroke develop deficiencies that prevent them from making reaching motions or doing activities of daily living (ADLs) such as eating, washing, and self-care. Therefore, a number of goal-oriented tasks and point-to-point reaching exercises are provided in rehabilitation to help the patients regain motor function and consequently their independence.

Typically, these therapies are administered by a therapist in a hand-over-hand manner in the clinic. As these tasks require a significant number of repetitions to be effective, it often uses up a major amount of the therapist's time and consequently only lets the therapist work with a few patients in a day. For this reason, rehabilitation robotics is an attractive replacement or complement to conventional physiotherapy. By programming a robot to assist patients with these tasks, more patients can be handled in the clinic as the robot would fill the position of a therapist.

While robots allow rehabilitation with minimal therapist intervention, low patient motivation is an issue. Added with the emotional impact of the disability, it is not uncommon to think the patients would lose their motivation through the rehabilitation process. A study shows that only 31% of users maintained their weekly exercise programs [66]. Due to this, video games have become a medium for robot-assisted rehabilitation therapy. Games provide an increase in motivation by making the task less of an exercise to endure but rather a more leisurely experience [67]. This means that there is an increase in motivation in patients using these technologies.

These games typically belong in one of two categories: Virtual Reality (VR) or Augmented Reality (AR). AR, in particular, has garnered attention in the past few years alongside immersive VR with the release of the Oculus Rift [68] and Microsoft's HoloLens [69]. With regards to rehabilitation, such technologies have proved to be effective in both physical and mental therapies [70][71][72].

3.1.1 Virtual Reality & Augmented Reality in Rehabilitation

Virtual Reality comes in many forms. They range from the non-immersive VR games that are displayed on a screen to immersive head-mounted displays (HMDs) such as the Oculus Rift. These games (mostly non-immersive) are now a common way to provide visual feedback about the therapy task during rehabilitation and are used by multiple rehabilitation robotics systems [58][73].

Augmented Reality, similar to VR, has different implementations. A Video See-Through (VST) AR setup lets the patient see overlays of digital images onto the video feed [55]. An Optical See-Through (OST) setup has the digital images calibrated to match what the user directly sees through a semi-transparent mirror [74]. Finally, spatial AR or projection removes the need for the user to wear any HMDs and allows direct interaction with the projected digital image [71]. The superimposition of digital images onto the real-world geometry is the common ground between the above-noted various AR implementations, allowing the users to feel that the digitally created objects are part of the real world rather than at a separate screen or in a virtual space. Therefore, unlike immersive VR, there is no need to render a virtual avatar of the user when using AR. Due to the direct interaction of AR, with proper calibration, colocation between the vision and the person's actions (can include haptic interaction with the digitally created object) can be achieved.

3.1.2 Haptic Feedback in Rehabilitation

Current rehabilitation devices in the commercial market come with VR games that focus on improving motor function. While effective, these games only provide the user with a visual (and possible auditory) feedback through a screen display. Therefore, the only way the users can know if they are doing the task properly is by seeing what their actions do on the game screen.

To add another level of interaction with the game, haptic feedback can be implemented. Haptic feedback stimulates the user's touch senses by reflecting forces to the user's hand. While commercial rehabilitation devices often do not have haptics implemented, this is a common method of improving patient experience in rehabilitation research. When combined with visual feedback, it can increase the realism of interaction and possibly the realism of rehabilitation outcomes. It can also open up various ways of assisting the users in terms of performing the task by tuning the feedback according to difficulty of the tasks.

3.1.3 Motivation for Visual-Haptic Colocation in Rehabilitation

Visual-haptic colocation is defined as the direct mapping of vision to physical interaction (including haptic interaction) between the user's hand controller and the object in the virtual scene. A majority of computer-integrated rehabilitation technologies that use robots to guide and assist users have them view computer games on a screen in order to be able to perform desired exercises. As there is a mismatch in axes of motion between the on-screen movements and the user arm's movement, the users must mentally "calibrate" themselves to map their arm movements to the coordinate frame of their avatar in the game (i.e., the users must imagine themselves positioned within the screen and move appropriately in the environment to complete the task). This causes the user to take a moment to do a mental transformation between the visual coordinate frame and their hand coordinate frame. However, the cognitive abilities of those suffering from disabilities may have also been affected negatively [75], which means they may have difficulty bridging the spatial disparity between the two coordinate frames. With a spatial AR setup, the visual and hand frames can be colocated, which could potentially lighten the mental load on the patient.

In this work, we will be incorporating haptic feedback in an AR setting in a rehabilitation environment in order to study the effectiveness of visual-haptic colocation. We will simulate a patient with cognitive deficiency by cognitively loading healthy participants in a user study. User task performance will be compared mainly between AR and VR for the different combinations of presence or absence of haptics and CL. The effect of cognitive loading will be briefly touched upon only to confirm if it does indeed significantly affect user performance in properly simulating patients. Each AR and VR pair will be analyzed to find which one benefited the most from visual-haptic colocation. The goal will be to help therapists and physicians find more efficient methods for rehabilitation. This investigation of bridging the spatial disparity could open up new possibilities for future rehabilitation games.

The chapter is organized as follows: Section 3.2 provides brief examples of work done in research literature. Section 3.3 describes the approach and game design. Section 3.4 outlines experimental setup, challenges, experimental procedure and a discussion of the results. Finally, Section 3.5 finishes with a conclusion and future work.

Literature Comparison					
	VR		AR		
	Non-Imm	Imm	VST	OST	Spatial
Non-Haptic	[76]	[77]	[78][79]	[80]	[71]
Haptic	[81]	[82]	[83]	[84]	[85][86] ●

Table 3.1: The table categorizes the different physical rehabilitation systems found in literature by the type of visual technique and incorporation of haptics. The black bullet represents this work's position in the literature.

3.2 Related Work

Multiple rehabilitation tasks have been published in the literature that fall within AR and VR as categorized in the above table. VR systems present a completely virtual environment to the user, allowing for creative scenarios unbounded by physical limitations found in real environments. AR systems let the user stay within their familiar environment while enabling interaction with the virtual objects that are displayed in the real world. A brief investigation of related work is presented to give insight on current technologies available for rehabilitation in research. These are categorized based on visual technique and the presence or absence of haptics to give a clear view of where this work stands.

3.2.1 Virtual Reality Rehabilitation Systems

VR systems often come in either non-immersive, or immersive displays. Non-immersive displays include 2D computer screens, TVs, or projection on a screen. The ReJoyce Rehabilitation Workstation is a non-immersive VR system with a multitude of interactive games to simulate a variety of ADL exercises [76]. However, it does not provide haptic feedback during the tasks, only visual and auditory. In another work, Adamovich et al. [81] presented a non-immersive haptic glove VR system to improve the hand function of post-stroke patients. Immersive VR systems typically use HMDs such as the Oculus Rift or HTC Vive. However, other systems may involve either the CAVE [87] or CAREN [88] systems that puts the user in a room with a large projected screen all around (CAVE) or in front (CAREN) of the user. In the work of Kaminer et al. [77], they used an Oculus Rift for an immersive VR pick-and-place task and used a non-haptic glove to record the hand's grabbing gesture. Likewise, Andaluz et al. [82] used the Oculus Rift and a Novint Falcon haptic device for their upper limb rehabilitation games.

3.2.2 Augmented Reality Rehabilitation Systems

AR systems are usually displayed using either VST, OST, or projection. While often portrayed as requiring HMDs, VST and OST can also be done using a monitor screen, however it lessens the immersion. For VST setups, Burke et al. [78] used a marker-based, non-haptic setup and developed a game similar to Atari's *Breakout* and another game where the participant stacks virtual objects onto a virtual shelf. Correa et al. [79] developed GenVirtual, a musical AR game where virtual cubes light up according to a musical sequence played and the user replicates the tune by occluding, with their hand, the colored cubes in the same sequence. Vidrios-Serrano et al. [83] used an HMD and a phantom Omni haptic device to touch virtual objects in the AR environment. For OST setups, Trojan et al. [80] took an non-haptic approach in developing a mirror training rehabilitation system suitable for home use. Luo et al. [84] used an HMD and a haptic glove for their AR hand opening rehabilitation setup. The glove was used to simulate holding a real object for their grasp-and-release task. For projection setups, Hondori et al. [71] created a non-haptic tabletop system for post-stroke hand rehabilitation which incorporated different games such as reaching a projected box to play sounds, holding a mug to pour virtual water, and grasping various sized circles. Finally, Khademi et al. [85] implemented a spatial AR setup and used a haptic device for monitoring the impedance of a human arm. They also did a comparison of AR vs VR for a pick-and-place task [86].

Technology Considerations

Other works such as the SITAR [89] also incorporated a tabletop workspace that uses an LCD television to visually colocate the patient's interactions with the tasks. Alongside with intelligent objects, the system can sense and provide haptic feedback. Despite having the option to choose a similar setup, we opted for a projection-based setup for a few reasons. While an LCD screen would be completely blocked off by an object above it, projection can still be seen above the placed object. There are also multiple ways to compensate for occlusion as will be discussed later. Furthermore, projection can adapt to different projection surfaces as with the case with projection mapping. This provides more potential for future studies in 3D AR viewing and dynamic interaction.

Other technologies such as the Microsoft Hololens, Oculus Rift, or HTC Vive were not considered since these devices are HMDs. For some patients, wearing an HMD may induce a sense of entrapment or anxiety from being disconnected from the real world [90]. Some works used haptic gloves for their systems. While gloves are effective in controlling finger flexion, extension and providing haptic signals around the hand, our work revolves around upper limb arm movements, therefore a haptic device or robot is more appropriate.

In our system, spatial AR, and haptics are combined in a rehabilitation task. While the work of Khademi et al. [85][86] provide similar investigations of performance between AR and VR, our focus lies in the use of cognitive loading to simulate similar cognitive behaviours (E.g. being distracted, inability to focus on one task) found in patients with reduced mental ability due to events such as stroke. It is understandable that this simulation may not fully capture the extent of damage a patient may experience from stroke, however, our aim is to determine if visual-haptic colocation is able to alleviate the negative effects of cognitive loading which can then be applied to actual patients in future discussions. By making the task easier for the patient to accomplish, the task success rate increases, thereby engaging the patient which results in a more effective rehabilitation.

3.3 Rehabilitation Game Design

Our aim is to set up a robot-assisted rehabilitation environment incorporating colocation of haptics and vision that would be both engaging and intuitive to use. With a 2D spatial AR system, we create an environment to perform reaching motions in which the end-effector of the rehabilitation robot needs to be manipulated by the user to push a digitally created car around a track. In order to create spatial AR, the image of the car and the track is updated in real-time and projected on the table supporting the robot. In this way, the haptic interaction between the user's arm and the robot end-effector happens in the same space where visual feedback about this interaction in the virtual environment is provided.

The goal of the game is to traverse a certain length of the track as soon as possible. The car can only be pushed from the back and cannot be moved in reverse. It is also constrained from moving sideways relative to the track. Upon collision of the end-effector with the car, force feedback is sent to the user along the contact normal. The track loops around the workspace and is composed of a Bezier spline. Every lap around the track, its curves changes slightly to keep the user stimulated and engaged. Only a limited portion – around 10% – of the track past the front of the car is seen and this is updated in real-time. The track could allow for clockwise or counterclockwise movement depending on the user's preference. The task is built in the Unity Game Engine Environment [91] and therefore uses the default Unity physics. This allows the car to have a momentum when pushed, giving the user the impression of pushing a toy car.

A blinking red dot on the corner of the projection is used as a visual aid when cognitively loading the users. An arithmetic cognitive task is simultaneously performed with the haptic task. Commonly used in gait and postural research [92], an articulated backward counting (multiples of 3) is chosen as it has been shown to be effective in decreasing performance when combined with another task. The constant subtraction is mentally taxing in that it requires continued attention in keeping track of the counted numbers to count down properly [93]. The dot turns on and off at a frequency of 1 Hz. The user then audibly counts down (starting from a randomized number between 100 - 200) every time the red dot appears.

Multiple conditions could be modified in the task. The absence or presence of force feedback, future track shape and the partial visibility of the track. Also, the cognitive loading could be altered by changing the frequency of counting down. From these, only force feedback is switched on and off between different runs of user trials and the rest are fixed at preset values.

We provide a training period to allow the participants to familiarize themselves with manipulating the end-effector and interacting with the car. When force feedback is turned off, the task becomes easier due to the removed resistance while pushing the car. Therefore, to allow for a fair comparison between haptics and no-haptics cases while keeping the difficulty of both scenarios on the same level, we resist the robot's movement with a damping-based controller when there is no force feedback. The amount of damping force applied is the same as the force felt when pushing the car when haptics is present. Haptic feedback is provided along the contact normal by a force that increases at a rate of 1.5N/s to a maximum value of 3 N. This is done to reduce contact instability from a sudden jump in force feedback at the time of contact.

The evaluation of user's performance is based on the time to traverse a fixed length of the track. For our user studies, we will be considering scenarios where force feedback, colocation of haptics and vision, and cognitive loading will be varied.

3.4 Experiment

The task was tried on 10 healthy participants with age range between 23 - 31 (9 out of 10 participants were males). The participants performed the experiments with their

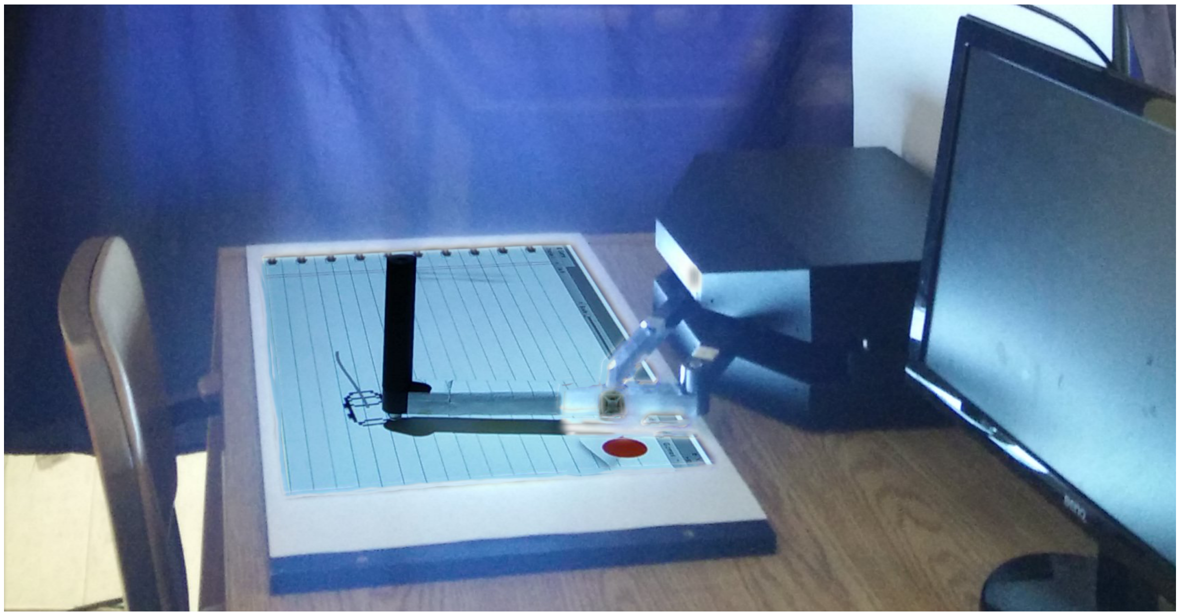


Figure 3.1: Side view of the experimental setup. The projector (not seen in the image) projects the task onto the table.

dominant hand (all right-handed) and were all from the University of Alberta community. 6 participants had prior experience with haptic interfaces. Each participant was provided with verbal instructions and was given a maximum of 5 minutes to familiarize themselves with the task.

3.4.1 Experimental Setup & Challenges

Our setup consists of an off-the-shelf InFocus IN116A projector and a 2-DOF planar rehabilitation robot (Quanser, Inc., Markham, Ontario, Canada). The projector is set directly above the projection space and the rehabilitation robot is positioned such that the end-effector can reach the majority of the projected area on a table. The task environment is created using the Unity Game Engine and the rehabilitation robot is controlled using MATLAB and Simulink. The experimental setup for the AR configuration can be seen in Fig. 3.1 and 3.2. For the VR setup, instead of displaying the game on the table using a projector, a computer monitor positioned in-front of the user on top of the rehabilitation robot is used.

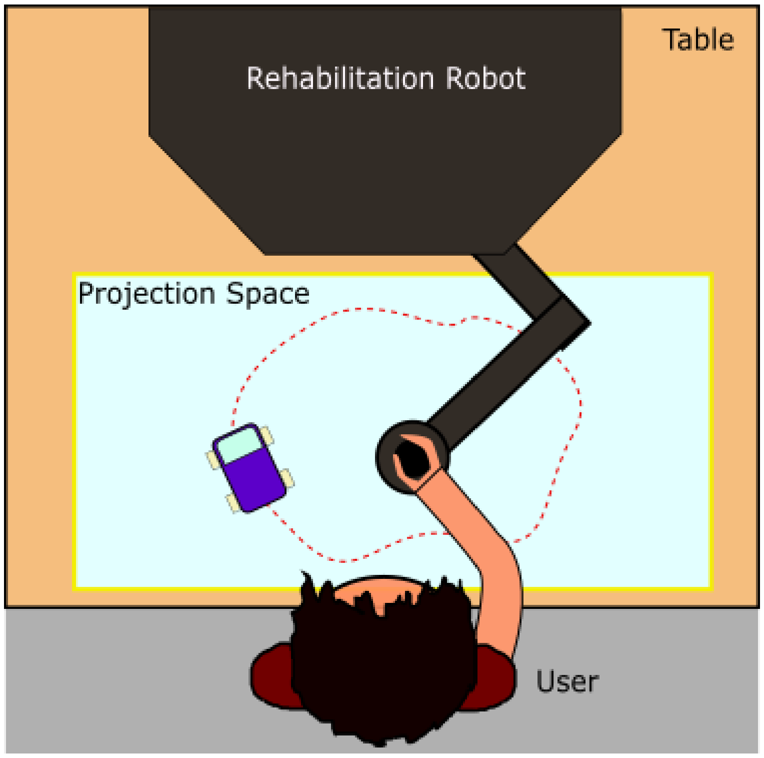


Figure 3.2: Top-down view of the experimental setup. The rehabilitation robot arm extends into the projection space to be used by the user to push the car around the track.

The development of the setup involves two main challenges that needed to be mitigated.

Occlusion

Due to the nature of projection, occlusion can be a big issue. The user's shadow, as they move around the track, could block the user from viewing the car properly. Typically, occlusion can be handled by having a depth camera and calculating the projector and camera intrinsics to project the virtual images such that the scene looks integrated in the user's environment from their viewpoint [94]. However, for this work, we do not intend to use the projection above the user's hand as a part of the game so a depth camera is not needed. Instead, there are a few modifications added to the task to minimize the effect of occlusion. The size of the car is increased so as to make it harder to lose the car and the workspace is elevated to minimize the distance between the robot's links and the table. The robot's links are also wrapped in white paper for better projection results. The visible portion of the track protruding from the car also allows the user to navigate easier.

Calibration

The task environment comprises a circular avatar that interacts with the car in order to move it around the track. To properly implement colocation, this circle must be projected exactly on the end-effector's position as the end-effector spans the workspace. To calibrate, four points are projected in a rectangular formation to the workspace and the end-effector position is recorded for each of the projected points. The point-to-point correspondence is done with a 2D projective transformation (homography) between the robot frame (hand frame) and the virtual frame [95]. Another method considered was affine transformations. However affine transformations, a subset of projective transforms, preserve parallelism. This would consequently amplify the errors if the end-effector placement on the projected points are inaccurate. On the other hand, projective transformations preserve only collinearity and incidence making it more general and can therefore compensate for any inaccuracies during calibration. Using the homography transformation H, each point-to-point correspondence i is mapped by the equation below:

$$\lambda p_i' = H p_i \tag{3.1}$$

where λ is a scaling factor, p and p' consist of the x and y coordinates of a point in the robot frame and screen frame respectively. An expanded version is shown below:

$$\lambda \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3.2}$$

When further expanded out as shown in the equations below, we can solve for x' and y' by dividing equations 3.3 and 3.4 by equation 3.5.

$$\lambda x' = h_{11}x + h_{12}y + h_{13} \tag{3.3}$$

$$\lambda y' = h_{21}x + h_{22}y + h_{23} \tag{3.4}$$

$$\lambda = h_{31}x + h_{32}y + h_{33} \tag{3.5}$$

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{3.6}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{3.7}$$

Equations 3.6 and 3.7 can be rearranged such that it is linear in terms of H.

$$-h_{11}x - h_{12}y - h_{13} + h_{31}xx' + h_{32}yx' + h_{33}x' = 0 \tag{3.8}$$

$$-h_{21}x - h_{22}y - h_{23} + h_{31}xy' + h_{32}yy' + h_{33}y' = 0 \tag{3.9}$$

The above equations can then be written in matrix form:

$$\begin{bmatrix} a_x \\ a_y \end{bmatrix} H = 0 \tag{3.10}$$

$$a_x = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \end{bmatrix} \tag{3.11}$$

$$a_y = \begin{bmatrix} 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \tag{3.12}$$

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{21} & h_{22} & h_{23} & h_{31} & h_{32} & h_{33} \end{bmatrix}^T \tag{3.13}$$

Since H is computed up to scale, we can impose a constraint $h_{33} = 1$. This makes the H matrix 8DOF. Each point provides 2 sets of equations, therefore a minimum of 4 points are required to get the homography transformation. Finally, we get an 8x9 matrix for A for which we use Singular Value Decomposition (SVD).

$$A = USV^T \tag{3.14}$$

U and V are 8x8 and 9x9 unitary matrices, respectively, and S is an 8x9 diagonal matrix whose elements are the singular values of A. The solution for H is then the last column of V, reconstructed into a 3x3 matrix. This transformation allows the circular avatar to be accurately aligned with the movements of the robot's end-effector. Once calibrated, this transformation is also used for the VR setup to ensure that the workspace for both AR and VR configurations are the same.

3.4.2 Procedure

A series of 8 different experimental conditions were presented to each participant. For each condition, the participant attempts to complete three laps around the track in the shortest time possible. The conditions defining each experimental trial involves

Task Performance Conditions			
		AR	VR
No CL	Haptic	1	2
	No Haptic	3	4
CL	Haptic	5	6
	No Haptic	7	8

Table 3.2: The table shows the conditions set for each task. Each set of conditions are labeled numerically. AR means the task is projected on the table and colocation is present. In VR, the task is done on a screen in front of the user where they use the rehabilitation robot as a joystick to control a circle to push the car. Note that these conditions are presented to the participants randomly to reduce the effect of learning.

a combination of the absence or presence of force feedback, colocation, and cognitive loading. The combinations of the task conditions are seen in Table 3.2. The order of presenting different conditions to a participant are randomized to minimize learning.

The experiment starts with the participant sitting at arm's length from the rehabilitation robot with the projection area in between them on the table. The system is then calibrated for colocation. The participant is also given a trial run with VR and AR (without CL but with haptics) to get an initial experience of the task. The series of 8 conditions are then presented as discussed before. Condition 1 is hypothesized to be the easiest as it provides to the participant both haptics and AR but no CL. Condition 8 is predicted to be the hardest task as it removes haptics, provides only VR and imposes CL to the participant. Therefore, it is expected that the results between Conditions 1 and 8 would have the biggest gap in user performance. After the experiments are done, the participants are given a questionnaire regarding their experience. This is to provide a subjective measurement of how engaged the participants were in either AR or VR. Ethics approval was approved by the University of Alberta Research Ethics & Management Online under the study ID MS9_Pro00033955.

Mean and SD in seconds of each set of conditions								
Cond. #	1	2	3	4	5	6	7	8
Mean	35.6	42.8	34.8	45.3	38.7	48.4	50.5	52.9
SD	6.0	7.3	9.0	12.6	5.0	6.2	12.8	13.7

Table 3.3: Table of Mean and SD of the time duration (seconds) results of all 10 subjects for each of the 8 conditions.

3.4.3 Results and Discussion

Duration Results

The participants were tested based on how fast they completed the trials. There is no penalty for missed counts during tasks with CL. The results are shown in Fig. 3.3. The mean and standard deviation (SD) of the results are reported in Table 3.3. To find the effects of each category on user performance, a 3-way RM ANOVA is utilized using SPSS 25 [96]. The 3 main fixed effects factors, colocation, haptics, and cognitive loading each present two levels. A correction method is required due to multiple comparisons. Therefore, for our post hoc analysis, the Bonferroni correction is chosen to reduce Type I errors. The ANOVA results report a significance in colocation $F(1, 9) = 8.773, p = 0.016$, marginally significant regarding haptics $F(1, 9) = 4.648, p = 0.059$, and significant in cognitive loading $F(1, 9) = 30.491, p = 0.000$

By conducting paired t-tests analysis with the Bonferroni correction only one notable pair is found statistically significant. In the simulated rehabilitation scenario when CL and haptics are present (Conditions 5 and 6), there is a significant difference in performance between AR and VR ($p = 0.0012$).

Haptics play a minor part in user performance without CL. When haptics is on, the participants overshoot as they miss the car while pushing. Turning off haptics lets the participants have better control when they miss the car, resulting in less overshoot. Since there is no CL, the participants notice the overshoot quicker and can correct their mistake faster.

As expected, trials that had CL resulted in longer times overall compared to those

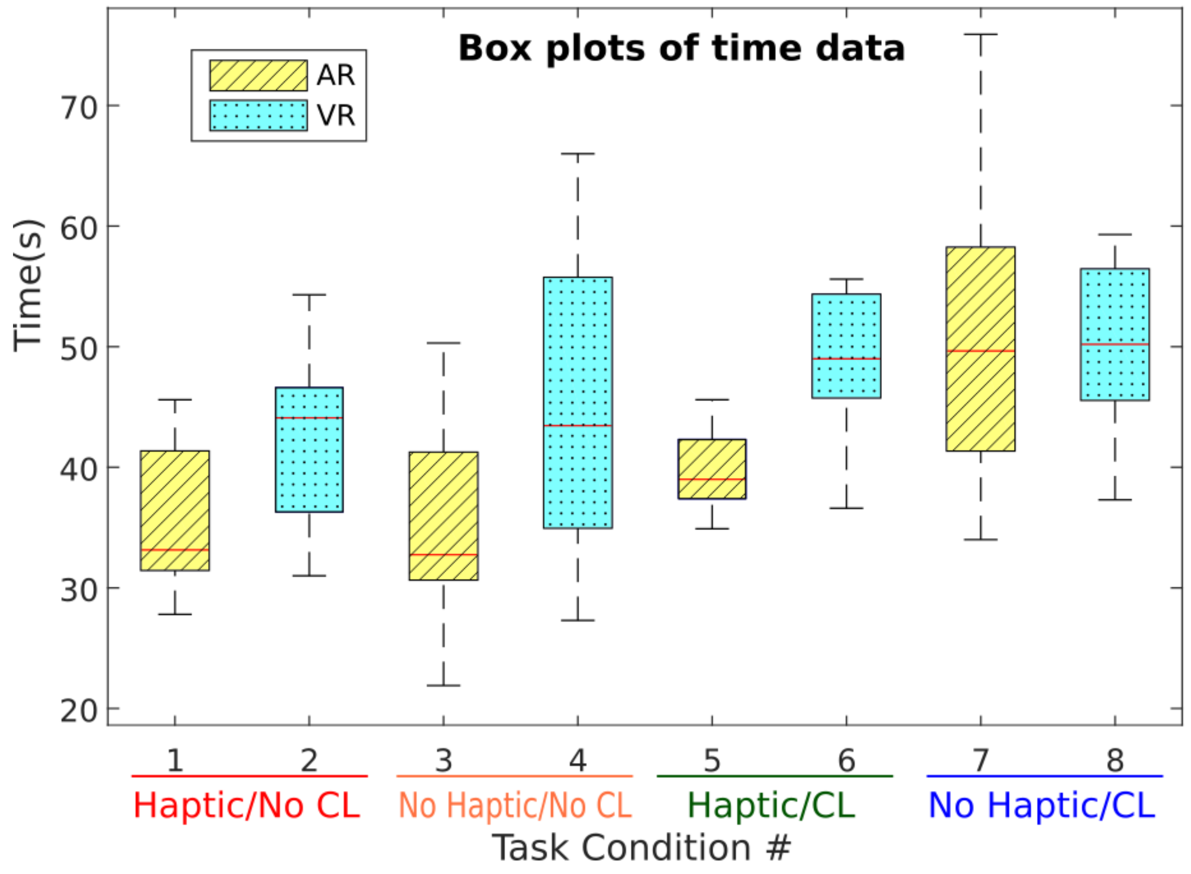


Figure 3.3: Collective user performance of 10 participants on the 8 different experimental conditions. The line within the boxes represent the median time.

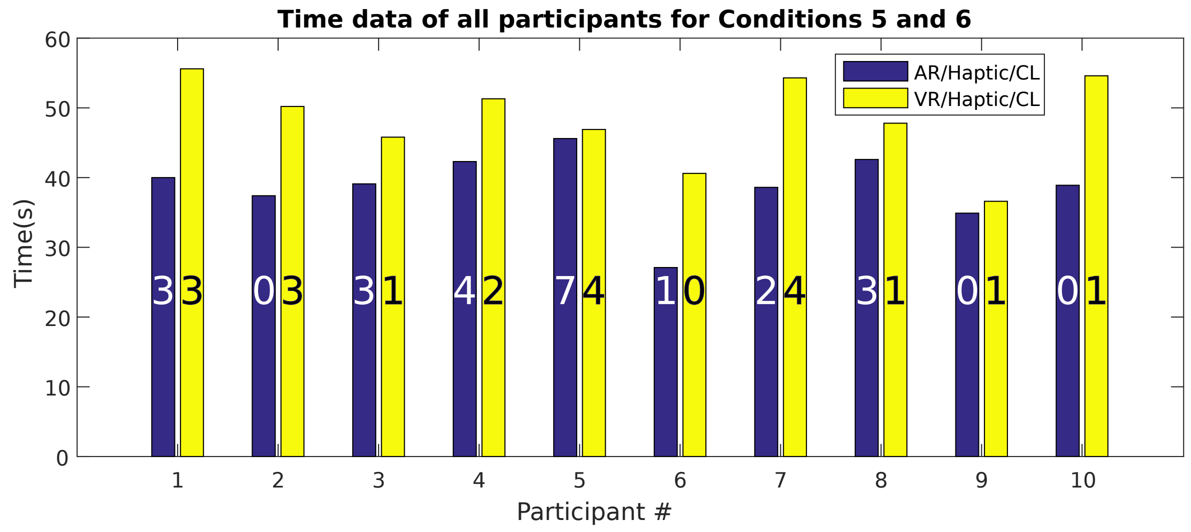


Figure 3.4: Time it took for each participant to complete Conditions 5 and 6. The number within the graphs represent cognitive loading misses.

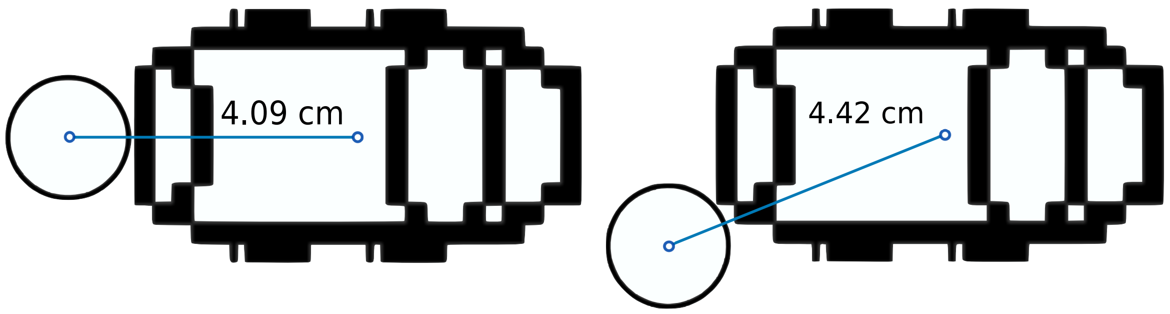


Figure 3.5: Allowable distance between end-effector and center of the car.

without it. AR and haptics with CL gave the best user performance out of the other CL trials. The participants mentioned that during CL and AR, they relied on haptics to keep the end effector behind the car since they were distracted by the CL. In the same scenario but without haptics, the participants would often take a while before realizing they lost the car and therefore taking a longer time to readjust. Focusing on the results of Conditions 5 and 6, Fig.3.4 shows that the participants had similar counts of CL misses even though Condition 6 resulted in longer times. It can then be assumed that the CL experienced by the participants are within similar levels. This leaves colocation to be the only differing factor between the two tasks. Therefore while presence of haptics is mostly irrelevant in other scenarios, for those undergoing CL, and hence those with their mental capabilities affected by disability, visual-haptic colocation is a favourable option.

Spatial Results

At each time instance, the robot's end-effector should be within a certain range to ensure it is in contact with the car. This means that the distance between the car center (its axis of rotation) and the end-effector should be between 4.09 *cm* and 4.42 *cm* as seen in Fig. 3.5. In Fig. 3.6 a sample of the car path and the user path for Conditions 5 and 6 is plotted for two participants. There is minimal direct overlap between the car path and user movement near the curved areas due to the distance

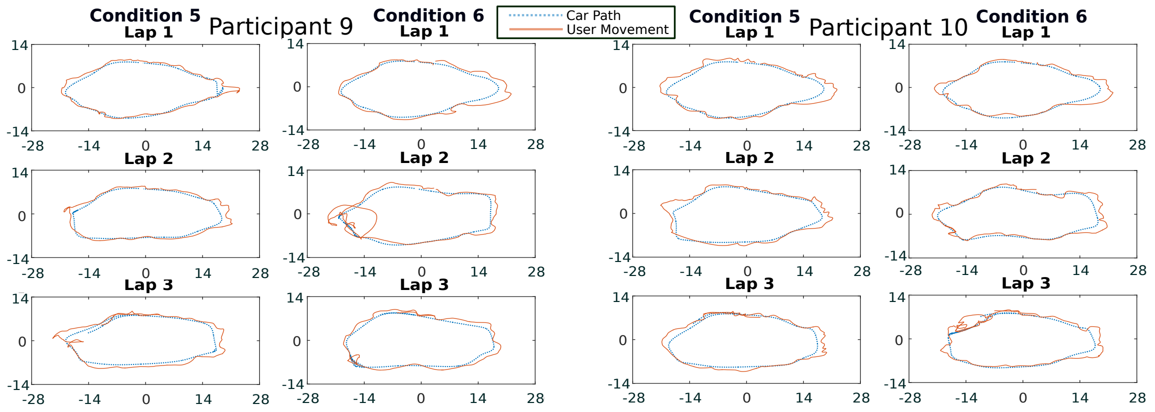


Figure 3.6: Snapshot of end-effector movement of two participants for both Conditions 5 and 6. Units are in *cm*. Both participants moved in a counter-clockwise fashion.

as previously mentioned. This distance is taken into account for the error calculation. Taking a look at a snapshot of their performance between the two task conditions, we can observe the participants missing the car and overshooting in Condition 6 for lap 2 and lap 3 respectively. The euclidean distance between the car and end-effector is calculated at each time sample and is subtracted from or by the minimum or maximum threshold respectively, depending on if the distance is below the minimum, or above the maximum. For participant 9, the average error for Conditions 5 and 6 is 0.52 *cm* and 0.97 *cm*, respectively. For participant 10 it is 0.09 *cm* and 0.61 *cm*, respectively. These further support the results found in Fig. 3.3 for the increased user performance when there is visual-haptic colocation during CL.

Survey Results

In the survey given after the experiments the participants were asked to rate their experiences in a 1-10 scale. The type of questions asked are shown in the survey results in Fig. 3.7. Likely due to the overshoot found in haptics, the easiness of the task in AR and without haptic feedback is rated slightly higher. Most of the participants however, rated haptic feedback to be more useful than without. This is reflected on the time results of Fig. 3.3 between task Conditions 5 and 7. The survey also shows that in all

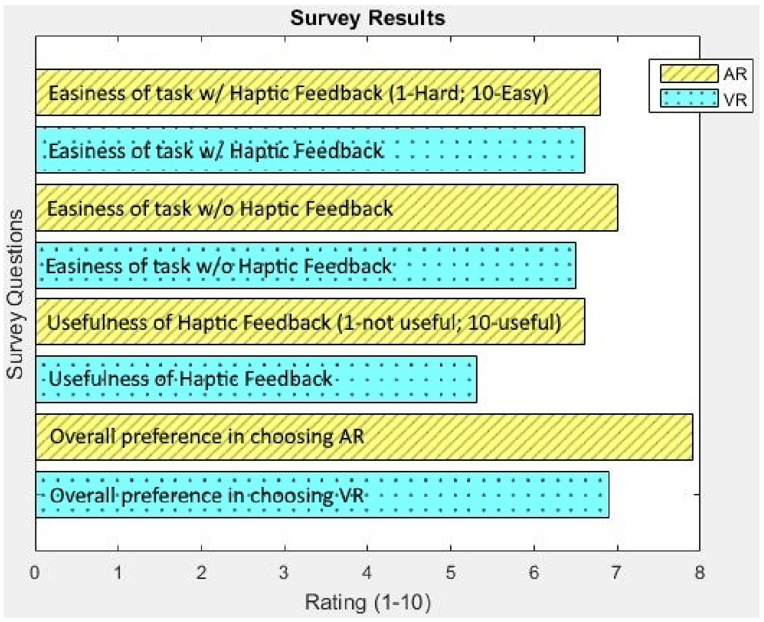


Figure 3.7: Survey given to the participants after the experiments. A higher number is a better rating

cases, AR was rated higher than VR, leading to AR being the preferred technology. A 5-point Likert scale is also included regarding the participants' experience when CL is present. This is used to match the task results with the perceived difficulty of the CL tasks. Specifically, in the range {-2, -1, 0, 1, 2} the participants are asked if CL made the task much easier (-2) or much harder (2). The resulting average is 1, indicating that participants thought CL made the task somewhat harder as can be seen with the increasing trend in time duration in the Fig. 3.3 results.

3.5 Conclusion

We present a comparison of the effects between colocated and non-colocated visual feedback when used in a rehabilitation environment. Visual feedback comes either in the form of spatial AR for visual-haptic colocation or non-immersive VR for visual-haptic non-colocation. The two are compared by measuring the performance of 10 healthy participants in a trajectory following task. To better simulate those with cognitive deficiencies, the participants are subjected to cognitive loading while performing the task. It is observed from the results that the effect of visual-haptic colocation improves the task performance, especially for those undergoing cognitive loading. For our future work, we plan to let patients with disability use the system and we will analyze the corresponding data. Other considerations would be to include an assist-as-needed functionality to help patients struggling with the task.

Chapter 4

A 3D Augmented Reality Display to Improve User Performance in Rehabilitation Exercises

4.1 Introduction

In recent years, rehabilitation has incorporated serious games using non-immersive virtual reality to motivate patients during therapy. Serious games are defined as video games designed for a purpose other than pure entertainment. Traditional rehabilitation training may involve training with real-world objects to do activities of daily living (ADLs) in order to help regain motor function. However, the repetitive nature of these exercises can make therapy a tedious process for the patients. Motivation is a key factor in predicting the success rate of rehabilitation [2][97]. In a study that involved post-stroke patients, only 31% of the patients maintained their weekly exercise programs [66]. For rehabilitation therapy involving serious games, it has been shown that there is an increase in motivation, providing patients with a more leisurely experience as they go through their therapy [67].

4.1.1 Virtual Reality & Augmented Reality Game Displays

The games are typically presented to the user in one of two forms: Virtual Reality (VR) or Augmented Reality (AR). Virtual Reality consists of a fabricated environment where the user controls an avatar or cursor to interact with the virtual world. This is often done through non-immersive VR that is displayed only on a typically flat 2D screen. However, with technologies such as the Oculus Rift and HTC Vive, immersive VR is becoming a more popular medium for serious games [82].

Augmented Reality is about the superimposition of digitally fabricated objects onto the real world environment. Applications for this can range from purely providing information or to allow interaction with the digital objects as if they existed alongside real world objects. AR consists of three main categories: Video See-Through (VST), Optical See-Through (OST), and Spatial AR (projection) [56]. VST utilizes a video feed where digital objects are overlaid onto the screen to interact with the real world objects [55]. OST overlays the digital images on a semi-transparent screen which allows the user to directly see the real world unlike VST [74]. Finally, Spatial AR projects the digital images directly onto the physical environment [71].

4.1.2 Visual and Motor Axes Colocation

While serious games are advantageous in enticing the patients to stick with their therapy program, they lack realism. The games need an interface which allows it to be controlled, such as a joystick or a haptic user interface. However, while using these interfaces, the game is typically displayed on a screen at a distance in front of the patient. However, in real-world tasks such as peg-in-the-hole insertion, the patients directly interact with objects, feeling and seeing them at the same location. In the games used for rehabilitation, however, there is a disconnect between the visual space and the movement space of the patient's arm. The mismatch between the axes of motion between on-screen movements and the patient arm movement require the patients to mentally "calibrate" themselves to map their arm movements to the coordinate frame of their avatar in the game. As such, due to the workspace mismatch between virtual and real worlds, the scaling of movements may also have to be accounted for. For those affected by events such as stroke that could have affected their cognitive processes negatively [98], doing a mental transformation between the visual coordinate frame and the hand coordinate frame could be a difficult task. The principal idea of this work is that to lighten the mental load on the patient and improve task success rates, the spatial disparity between the coordinate frames can be bridged using AR.

Two hypotheses are investigated:

1. *Regardless of the presence or absence of cognitive loading, AR improves user performance over VR.*

2. *The results of AR during cognitive loading is not significantly different from AR without cognitive loading.*

The focus of this work is only to show that AR can make it easier for the patient to perform the task, thereby increasing the likelihood of success in performing that task. Actual motor scores comparing improvements of AR against VR will not be shown. That would require a longitudinal study on a treatment group and a control group of patients who would come into the clinic for at least 3 months every week, 3 times a week. In such a study, treatment group would be receiving AR, while the control group receives AR. Standardized assessments such as Fugl-Meyer would be used to compare the scores. This will not be included in this chapter but is instead future work.

4.1.3 Related Work

Most rehabilitation literature that incorporate serious games involve 2D non-immersive VR implementation, or AR in 2D or 3D but without colocation of visual and motor axes. Devices such as the ReJoyce Rehabilitation Workstation have multiple interactive 2D games to motivate patients and improve upper limb function after stroke [58][76]. Correa et al. [79] created a musical AR game called GenVirtual which is a spatial 2D AR game where the user replicates the tune produced by virtual cubes that light up in a sequence by touching the cubes in the same order. Gama et al. [99] developed MirrARbilitation, a VST 2D non-colocated AR system to encourage and guide users in a shoulder abduction therapy exercise.

For the case of 3D, Vidrios-Serrano et al. [83] used a VST 3D non-colocated AR system integrated with a phantom Omni device to interact with the virtual environment in a rehabilitation exercise. Broeren et al. [100] and Murphy et al. [101] used a haptic immersive workbench to test both able-bodied and stroke-impaired persons for rehabilitation and assessment with their OST 3D colocated AR system. Swapp et al. [102] studied the effectiveness of a 3D stereoscopic display AR system for colocated haptic feedback. Swapp examined if there is a benefit in having visual-haptic colocation as opposed to not having it. However, the study did not look at its effects in rehabilitation exercises and did not adjust the display as the user moved his/her head. Unlike Swapp's work, we hope to show that by using a 3D AR system, even users who are unable to perform with their full mental capacity are able to show improvement over the 2D system.

The difference between ***visual-haptic*** colocation and ***visual-motor*** colocation

In Chapter 3, we looked into the advantage of visual-haptic colocation (visual and motor axes colocated with the addition of haptic feedback) in a 2D environment. The term *visual-haptic* colocation was used as opposed to *visual-motor* colocation since we directly altered the presence and absence of haptics to study its effects alongside colocation. Different conditions were tested, which included haptics being present even when the visual and the motor axes were not colocated. Visual-haptic colocation simply indicated that all three (visual, motor, and haptic) axes were aligned. In this chapter, since haptic feedback will always be present in each task regardless of visual and motor axes alignment, visual-motor colocation is used instead.

Why 3D?

The findings in the previous chapter showed that in 2D, AR generally produced better user performance than VR. To further expand on this study, we chose to continue and investigate this in a 3D environment as well. 3D visualization of tasks and their comparison with their 2D counterpart has been performed in other areas of research such as surgery and has been found to increase surgical efficiency [103][104]. Applying the same concept to rehabilitation exercises allows us to examine ways to make therapy exercises easier and more effective for patients. Furthermore, 3D visualization of rehabilitation tasks brings the experience closer to a traditional rehabilitation exercise in which patients manipulate real-world objects. Since the state of the art in rehabilitation robotics mainly consists of 2D VR systems, it would be the ideal reference point for comparing the differences of a 3D AR system.

This chapter investigates, using a 3D Spatial AR setup, the effectiveness of 3D AR Visual and Motor Axes Colocation compared to 2D non-immersive VR in a rehabilitation context. User task performance is compared between AR and VR cases. A patient with cognitive deficiency will be simulated by cognitively loading able-bodied participants. We expect that if AR is able to make differences in performance for able-bodied participants who are distracted by cognitive loading, then it is possible to see such differences in actual cognitively-challenged patients as well.

The chapter is organized as follows: Section 4.2 describes the tasks for the proposed system and the experimental setup. Section 4.3 explains how the experiments are carried out and provides a discussion of the results. Section 4.4 concludes the chapter by summarizing the work and findings.

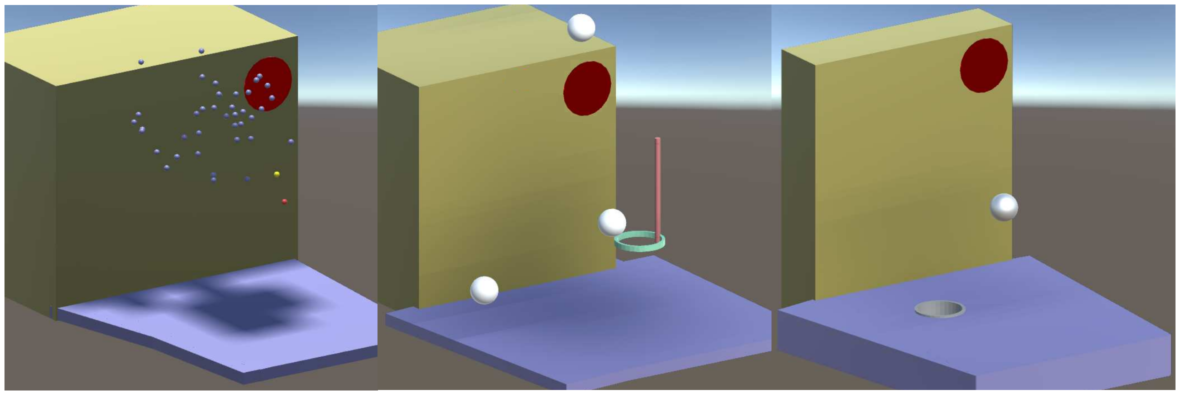


Figure 4.1: The three tasks, *Snapping* (left), *Catching* (centre), *Ball Dropping* (right).

4.2 Proposed 3D Spatial AR System

A 3D AR rehabilitation environment should have the same elements of a traditional rehabilitation environment but with the flexibility and creativity that a virtual environment can bring. The proposed system involves visual and motor axes colocation and depth perception to immerse the user in the task. Three tasks are created to test the user performance between 3D AR Visual-Motor Colocation and 2D non-immersive, non-colocated VR: *Snapping, Catching, and Ball Dropping.* The tasks can be seen in Fig. 4.1.

4.2.1 Representative Tasks

The *Snapping* task requires spatial awareness and accuracy. The user controls a small ball and manipulates it around 40 other small spheres. At any given moment, only one of the spheres will be highlighted to indicate the target position for the ball. The user has to move the ball to the location of the highlighted sphere. When the ball and the target sphere overlap, the end-effector holding the ball will snap onto the overlapped sphere letting the user know via haptics that they succeeded. This prompts a new sphere to be highlighted, which the user has to get to next. Each highlighted sphere that is reached scores a point. Along the way, collision with the unhighlighted spheres must be avoided since it will reduce the score by one for each point hit. Overshooting a highlighted sphere that was just hit (thus becoming unhighlighted) and coming back to also subtracts one score. This encourages the users to maintain a balance between speed and accuracy throughout the 60s the task is run. If the user is able reach all 40 spheres, the first sphere becomes highlighted again and the exercise continues.

The *Catching* task tests the user's performance with manipulating the end-effector in a fast-paced scenario. The task is to catch balls that fall from a ledge using a hoop attached to a stick which is controlled by the end-effector. This requires the user to reach around the workspace, have good reaction time, and have spatial positioning accuracy to catch the balls. The balls spawn above the ledge at random locations every 2 seconds and come towards the user at different speeds; therefore, they fall to different areas of the workspace depending on their speed. Each time a ball enters the hoop successfully, the user scores a point. The task runs for 60s, allowing for 30 balls in total to be spawned. Whenever the balls hit the edge of the hoop, the user feels confirmatory haptic feedback on the end-effector.

The *Ball Dropping* task requires precision and accuracy. A hole is spawned at a random position on the desk surface. The user controls a ball that is positioned approximately shoulder height of the person and aims it above the hole. The ball is released by pressing the spacebar on the keyboard. The location of the hole changes as soon as the ball goes through the hole and the ball returns to the user. Otherwise, the user will have to try dropping the ball into the same hole until it successfully goes in. The task runs for 60s, giving the user a point for every time the ball falls through the target hole. There is no score limit for the number of balls successfully dropped in the hole.

For each of the tasks, a red dot blinking at 1 Hz is placed on a virtual wall across from the user. This is the visual aid that is used to keep the users in tempo during cognitive loading. To simulate cognitive deficiency in able-bodied participants, an arithmetic operation is done by the participants alongside each of the three previously mentioned tasks. Counting backwards in multiples of 3 has been shown to be effective in decreasing user performance during dual task performance [92][93]. A random number between 100 - 200 is given to the participants before the start of each task. Every instance the red dot appears, the participant is to audibly count down, constantly subtracting by 3 each time.

4.2.2 Experimental Setup

The system uses a High Definition Haptic Device (HD^2) from Quanser, Inc., Markham, Ontario, Canada. The HD^2 device is used as the interface to interact with the digital objects. An off-the-shelf InFocus IN116A projector is mounted above on the wall behind the user. It projects the task on the curved screen similar to [105] that is 65 *cm* tall, 85 *cm* deep, 56 *cm* wide. Similar to the work of Swapp et al. [102], a television or monitor could have been chosen as the display medium. However, a projector was chosen due to its versatility. A projection setup will be able to keep our options open for future work for larger scale exercises that may involve walking, or even users moving around in a wheelchair. With the depth information provided by the Kinect, a model of the scene can be constructed and projected on to such that from the user's viewpoint, the virtual object is displayed properly. This is achieved using the RoomAliveToolkit [106] even though the projection surface is not flat. Being able to use two perpendicular surfaces as the display allows for more creative tasks and better immersion for the user. Our setup uses a curved screen to have a seamless projection surface. Having no corners or creases in the projection improves the experience of the user [105][107].

A Kinect V2 sensor is located above the screen facing the user for head tracking purposes. Head tracking is crucial to the setup of the system to allow patients to have the freedom of movement while still having proper perspective on the virtual environment. Otherwise, they would need to keep their head positioned on the same spot during the entire exercise in order to keep the correct perspective. The HD^2 is located on the right side of the screen such that the end-effector can be moved around the centroid of the curved screen. The HD^2 was used to interact with the virtual

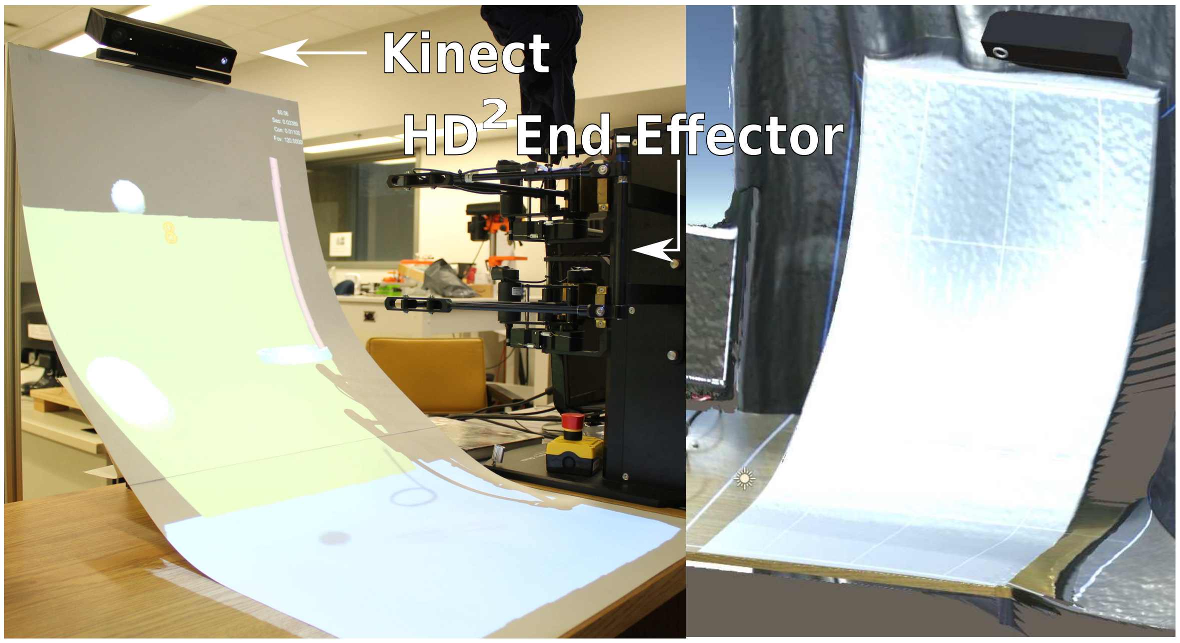


Figure 4.2: Left: Actual setup. Task is projected onto the screen (projector is not in view). Right: Model of the setup created in Unity.

environment rather than utilizing the Kinect depth sensor to interact using freehand motions. Freehand lacked the haptic functionality that robotic devices have. The haptics add another layer of feedback with the virtual environment, adding realism to the interaction for better immersion.

The task workspace where interaction with digital objects occur is in the space between the screen and the user. The task is created using the Unity Game Engine [91] and utilizes the open-sourced RoomAliveToolkit [106] to handle the kinect-projector calibration. The HD^2 is controlled using MATLAB and Simulink. The experimental setup can be seen in Fig 4.2. The virtual camera within Unity, which provides the view of the virtual environment, is positioned such that the virtual environment is integrated in the real environment when seen from the user's viewpoint. This camera acts as the user's eyes and is displaced left and right at 60Hz to enable stereo-viewing for 3D depth perception. The user then wears active DLP-link 3D shutter glasses to properly see the environment in 3D.

Calibration

To match the HD^2's end-effector movement with the virtual environment, the position of the HD^2 only needed to be aligned to match the position of the user-controlled virtual object. The HD^2 encoders provided accurate readings for the end-effector and Microsoft's RoomAliveToolkit scaled the Unity environment to match the real-world. In the same vein, head-tracking with the Kinect is also aligned such that both virtual and real world environments are matched in scale. This is to ensure that regardless of the position and angle it is viewed from, the virtual world would seem part of the real world.

Occlusion

As with any projection systems, occlusion is an issue when objects create a shadow that blocks the projection onto the screen and instead the images gets projected on the object (e.g. on a user's hand). Likewise, improper rendering of the virtual objects when the projection surface changes reduces the immersion of the user with the virtual environment. Positioning the virtual objects that the user controls directly onto the user's hand breaks this immersion. In light of this, the virtual objects are positioned at a small offset (1 cm) to the left and back of the end-effector to prevent improper rendering and shadow occlusion. This is also done to prevent the end-effector of the HD^2 from occluding the controlled object. The virtual environment is also displayed such that the majority of the workspace the user interacts with is projected on the upper area of the screen to reduce shadow occlusions. The head tracking carried out by the kinect also helps the user look around objects in the case occlusion occurs.

4.3 Experiment

A total of 10 able-bodied participants (ages 22-32) from the University of Alberta community took part in the experiments. All participants were right-handed and had

Task Performance Conditions		
	VR	AR
No Cognitive Loading	1	2
Cognitive Loading	3	4

Table 4.1: The table shows how each task is split into 4 conditions. While there are 4 conditions, each condition is presented twice to the participant to increase the validity of the results. To summarize, there are 3 tasks, 4 conditions/task, 2 trials/condition to give a total of 24 trials. Note that the numbering on the table is only for reference for the other figures in this chapter and does not reflect the order the conditions are presented to the participants.

prior experience with haptic devices. 5 out of 10 had experience with using shutter glasses or VR headsets. Verbal instructions were provided alongside a trial run for each task for familiarization.

4.3.1 Procedure

Each participant is presented with three tasks: *Snapping, Catching, and Ball Dropping.* The order of presentation is randomized to prevent any bias in learning effects happening between tasks. Within each task, two independent parameters are manipulated. Each task is done in either 3D AR or 2D non-immersive VR. The presence of cognitive loading is also switched on and off. Haptic feedback is turned on for all trials. Thus, there are 4 conditions to be tried for each task. Each condition is presented twice to the user and is given in random order. Since the random generation of locations for each task may bias the results (e.g., the balls in the catching task might spawn in similar locations/speed for one participant, but far apart for another), two sets of spawn points are generated for each task. Therefore, each participant attempts each of the three tasks 8 times, giving a total of 24 trials per participant.

Each participant is seated at arms length from the projector screen with the HD^2 to their right side, giving the end-effector access to the area between the projector and the participant. While they are wearing the shutter glasses, the eye separation is then measured by having the participant compare a virtual end-effector with the

Mean and Standard Deviation of Outcome Measures of Each Task				
Cond. #	**1**	**2**	**3**	**4**
		Snapping Task		
Total Score	23.0 ± 7.7	27.0 ± 10.6	17.9 ± 5.3	23.2 ± 9.0
Wrong Hits	6.5 ± 4.3	5.3 ± 4.5	3.8 ± 2.3	2.5 ± 1.7
Net Score	16.6 ± 4.9	21.7 ± 10.7	14.1 ± 3.8	20.7 ± 8.2
Time/point	1.8 ± 1.2	1.5 ± 1.2	2.2 ± 1.5	1.7 ± 1.4
		Catching Task		
Score	6.9 ± 2.1	14.2 ± 2.9	3.7 ± 1.6	9.5 ± 3.4
		Ball Dropping Task		
Score	7.9 ± 2.2	11.0 ± 3.6	4.9 ± 2.3	8.9 ± 2.9
Tries/Hole	3.2 ± 1.0	2.2 ± 0.7	6.0 ± 3.7	2.5 ± 0.9
Time/Hole	8.5 ± 1.9	5.7 ± 1.6	16.3 ± 11.2	6.6 ± 2.7

Table 4.2: Table of mean and standard deviations for the outcome measures for each of the three tasks. Results show the average per person.

HD^2's end-effector. The separation is adjusted until the virtual end-effector is parallel to the HD^2's. Then in random order, the sets of tasks are presented, each with 8 trials that are also randomized. Before the start of each new task set, a trial run is given in 3D AR for the participants to get a feel of the 3D environment and the task. Overall, the experiment lasted for approximately an hour and fifteen minutes per participant, including the resting time between each task. This study was done with approval from the University of Alberta Research Ethics & Management Online, ID MS9_Pro00033955.

4.3.2 Results and Discussion

No penalty is applied to cognitive loading miscounts. Since each of the 4 conditions are presented to each participant twice, the results present the average of the two trials. A Kolmogorov-Smirnov test [108] for normality was done for the net score of the snapping task, and both score results of the catching and ball dropping task. All three passed the normality test ($p > 0.05$) thus accepting the null hypothesis that the data comes from a normally distributed population. A 2-way Repeated Measures

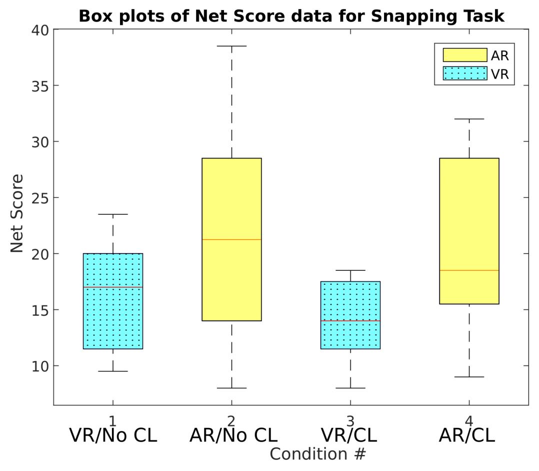
Analysis of Variance (RMANOVA) [109] is applied to the results to determine if there is a significant difference between the results for the different conditions. The main fixed effects are the visual techniques used (AR or VR) and cognitive loading (on or off). The False Discovery Rate (FDR) correction is chosen to reduce Type I errors for our post-hoc analysis [110]. A Type I error is also known as a "false positive" result, which is the rejection of a true null hypothesis (i.e., mistakenly accepting that the data is statistically significant when it is not). In the box plots in Fig. 4.3, the significance is represented by the stars (*) on the horizontal line above two sets of conditions; One star (*) represents a significance value of $p < 0.05$ and two stars (**) represent a significance value of $p < 0.01$. If there is no horizontal line above two the results for conditions, there is no statistical significance between them.

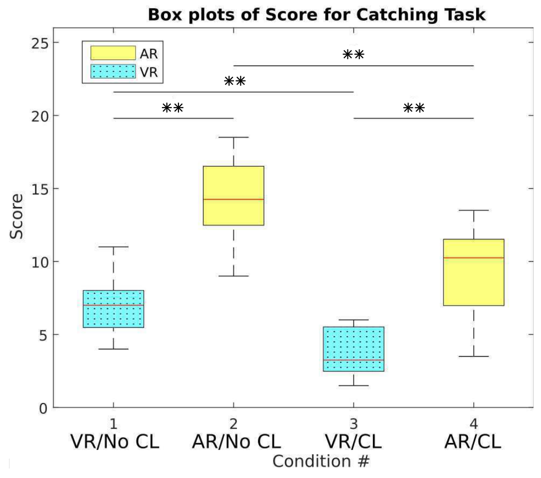
Score Results

The mean and standard deviation (SD) results of each task and its conditions are shown in Table 4.2. For the Snapping task, four areas of scoring were collected: Total Score - the number of highlighted points reached, Wrong Hits - the number of unhighlighted points hit, Net Score - the final result after subtracting wrong hits from the total score, and Time/point - the amount of time it took to travel between points. The Catching task only includes the amount of times a ball is successfully caught in the hoop, denoted as the score. The Ball Dropping task has three outcome measures: Score - the number of balls successfully dropped into the hole, Tries/Hole - the number of attempts the participants had to try before successfully getting the ball in, Time/Hole - the time it took, in seconds, before the ball went in.

Statistical Significance between Conditions

As seen in the box plots, the scores for AR are generally higher than VR for all tasks and all cognitive loading conditions. RMANOVA results in Table 4.3 show significance in all measures for both fixed effects except for two from the Snapping task.





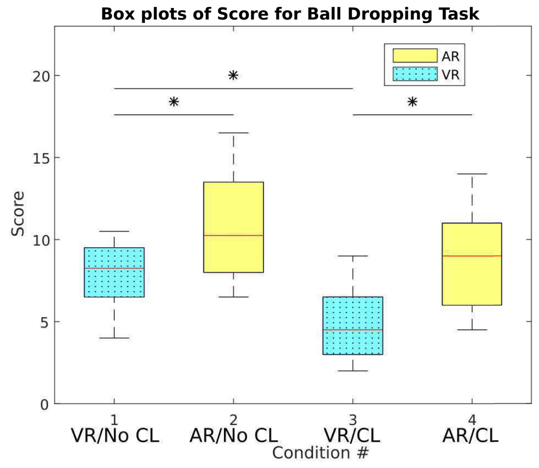


Figure 4.3: Box plot results for Snapping (Top), Catching (Middle), and Ball Dropping (Bottom). The line within the boxes represent the median score. The horizontal line above the conditions show the statistical significance of the two conditions. One star (*): $p < 0.05$. Two stars (**): $p < 0.01$. No horizontal line represents no statistical significance.

RMANOVA Results			
Task	**Measure**	**VR vs. AR**	**No CL vs. CL**
Snapping	Total Score	**F= 6.4, p= .0321**	***F= 22.0, p= .0011***
	Wrong Hits	$F = 3.4, p = .1004$	**F= 8.3, p= .0179**
	Net Score	**F= 6.7, p= .0291**	$F = 3.0, p = .1178$
Catching	Score	***F= 46.7, p= .0000***	***F= 103.3, p= .0000***
Ball Dropping	Score	***F= 38.3, p= .0002***	***F= 14.1, p= .0045***
	Tries/Hole	***F= 12.1, p= .0069***	**F= 8.1, p= .0194**
	Time/Hole	***F= 10.9, p= .0093***	**F= 8.0, p= .0196**

Table 4.3: The table shows the RMANOVA results of each task category for each main fixed effects (VR vs. AR and No CL vs. CL). The F-ratio and p-values are reported. **Bolded** values represent $p < 0.05$ significance. ***Bolded italicized*** values represent $p < 0.01$ significance.

A paired t-test with the FDR correction, as seen in Table 4.4, is utilized to closely inspect if there are significant differences between the conditions. While the RMANOVA results for the three measures in the Snapping task showed significance in some of the main fixed effects, t-test results show a lack of significant difference between the condition pairs. From observation in both the data in Table 4.2 and during experiments, participants moved between points faster when there was no cognitive loading. Consequently, the gross number of points reached (total score) was much higher than their CL counterpart. However, this also caused a larger amount of unhighlighted points hit. With CL, the participants took longer and acted more carefully, therefore colliding less. This provided an unintended result with cognitive loading that does not reflect what is perceived to be a patient with cognitive deficiency and therefore no conclusions can be made with the analysis for this task.

For the Catching task, RMANOVA shows a statistically significant difference between the presence and absence of both visual-haptic axes colocation and cognitive loading. Paired t-tests show that our first hypothesis is met; AR resulted in better success scores compared to VR regardless of the presence or absence of CL. However, the second hypothesis is not met. For this task, visual-motor colocation via AR enhanced user performance over VR, however it did not greatly improve it such that the

Paired T-test p-value results between Conditions				
Condition Pairs	**1 vs 2**	**3 vs 4**	**1 vs 3**	**2 vs 4**
Snapping Task				
Score	0.3929	0.2531	0.2531	0.3929
Wrong Hits	0.5644	0.2236	0.2047	0.2047
Net Score	0.2913	0.1325	0.2913	0.8081
Catching Task				
Score	***0.0000***	***0.0003***	***0.0016***	***0.0041***
Ball Dropping Task				
Score	**0.0425**	**0.0129**	**0.0168**	0.1808
Tries/Hole	**0.0415**	**0.0415**	**0.0481**	0.3779
Time/Hole	***0.0072***	**0.0321**	0.0600	0.3513

Table 4.4: Table of Paired T-test results between two conditions using the False Discovery Rate correction. **Bolded** values represent $p < 0.05$ significance. ***Bolded italicized*** values represent $p < 0.01$ significance.

CL case would produce similar results with the non-CL case.

For the Ball Dropping task, there is statistical significance between the average scores, average tries per hole and the average time the participants took per hole as seen in the RMANOVA results. This suggests that for AR, participants spend less time and are more confident in how they position the end-effector. Mostly seen in the VR case, the participants also utilized the ball's shadow to gain depth information. According to RMANOVA, cognitive loading produced a significant effect in decreasing user performance. However, paired T-tests show that this significance is more prominent between VR results. Therefore, both hypothesis 1 and 2 have been met. AR in both non-CL and CL case had significant improvements over VR, but under AR, there was no significant difference between CL cases. This shows that AR was able to alleviate the negative effects of CL.

Observations

The three tasks tested spatial manipulation, accuracy, and awareness. Each task also differed in regards to how dynamic the participants had to be with their movements, speed, and reaction time.

The Catching task, for instance, consistently required fast movements to catch the balls while the Ball Dropping task allowed the participants to take their time in determining the positioning required to successfully drop the ball in the hole. Effects of cognitive loading also varied between the three tasks. CL had the biggest influence in decreasing user performance in the Catching task. This is mostly due to requiring fast movements and reaction time while simultaneously undergoing CL. Participants would often slow down their movements while thinking of the next number.

In the Ball Dropping task, participants counted down in sync with the moment they press the spacebar to drop the ball while expecting the ball to fall in. When it does not go in, they press the spacebar again to retrieve the ball as quickly as possible, causing a break in concentration during CL and thus slowing them down. Its effect in VR compared to AR is much greater due to the lack of depth perception, requiring more tries before succeeding.

For the Snapping task, while the paired t-test fell short of providing a significant difference, the box plots portray hints of improvement in the AR cases. Due to the task not requiring reaction time like in the Catching task, nor anticipation of success as seen in the Ball Dropping task, participants in the Snapping task moved more carefully and steadily when CL was applied. They also moved in sync with their counting, snapping onto the target points during each countdown. These factors may have contributed to our varied score results.

Feedback from the participants came in the form of verbal comments and certain habits noticed while the tasks were being done. All the participants made use of the head tracking to view the environment from different angles for better depth information. This was more evident in the snapping task which needed spatial awareness of the surroundings to avoid the unhighlighted points. Halfway the 24-trial point of the experiments, a few of the participants became accustomed to the backwards counting. Two of them suggested different ways of providing cognitive loading such that it is variable, making it harder to get used to. While the learning is evident in prolonged trials, the randomization of the trials aided in reducing its effect. Participants with experience in either immersive VR or AR technologies adapted faster to the tasks. Those without experience often needed more time, in the earlier trials, to adjust their eyes to the AR environment.

4.4 Conclusion

Comparisons are performed in user task performance between 2D non-immersive VR and 3D spatial colocated AR. We showed that by bridging the gap between visual coordinate frame and hand coordinate frame, able-bodied participants with a simulated cognitive deficiency will experience improved success rates in the rehabilitation exercises. Since disabling events such as stroke affects the central nervous system and therefore possibly causing cognitive disability, we simulate this cognitive disability through cognitive loading in the form of an arithmetic operation. Three tasks were presented to the participants: Snapping, Catching, and Ball Dropping. In terms of superiority of performance in AR over VR, the main hypothesis of this work was not met in the Snapping task. The Catching task met the requirements of the main hypothesis; AR proved to significantly enhance user performance in both non-CL and CL cases. The Ball Dropping task also confirmed the first hypothesis, but further improved the success rate of AR during CL, therefore meeting the requirements of the second hypothesis in which the negative effects of CL are alleviated to allow the user performance of AR during CL to not be significantly different from the non-CL AR case. Future work include testing the system on actual patients as part of a longitudinal study. Incorporating the system with assistive functionality to further improve success rate is also considered. By introducing the benefits of visual-motor colocation in a 3D augmented reality rehabilitation system, we hope to inspire new possibilities of rehabilitation games that are not bound by the limits of 2D monitors.

Chapter 5

A Robotic System with an Augmented-Reality Display for Functional Capacity Evaluation and Rehabilitation of Injured Workers

5.1 Introduction

The growing demand for rehabilitation services following a workplace injury has motivated the development of new technologies for robotics-assisted assessment and rehabilitation of motor function following injury. The standard practice in occupational (or vocational) rehabilitation is to first perform a functional assessment of the injured worker. Typically, this is done using a Functional Capacity Evaluation (FCE) that assesses a worker's performance in a set of standard tasks [111], where each task requires different sets of equipment. The tasks incorporated in the FCE may involve material-handling activities such as lifting, pushing, and pulling, and positional tolerance activities such as walking, reaching, and grasping [112].

The first problem with the above is that it needs a large amount of equipment for various functional tasks and the space to store them. While a small number of all-in-one computer-based assessment tools exist [113][114], they are highly specialized in design and can replicate only specific rehabilitation tasks. A second problem emerges due to the current standardized assessments, where therapists qualitatively assess a patient's performance based on what they can observe. More complex, quantitative and objective assessments are desired. A third problem occurs when therapists increase the difficulty of a task or ask the injured workers to execute tasks that are considered boring; the patients can become bored, unmotivated, or uncooperative.

To address the above issues, we propose a generalized robotics-based solution. Our solution incorporates a serial-manipulator and a projection-based Augmented Reality (AR) display in order to provide a unified tool for both FCE and rehabilitation that is immersive and device-independent. To evaluate the efficacy of the proposed system, the biomechanics of the user's arm while using the system is retrieved and compared against the biomechanics of their arm in an equivalent real-life performance of the same task. In this regard, we present the following hypothesis: The proposed system can be used as an alternative to traditional occupational rehabilitation exercise environments because *it does not significantly modify the biomechanics of the user's arm while performing functional tasks compared to the conventional task performance.*

The chapter is structured as follows: Section 5.2 is a brief overview of the work found in the literature that relates to our proposed approach. Section 5.3 describes the design of the rehabilitation exercise and experimental procedure. Section 5.4 presents the results and provides a discussion based on the performed data analysis. Finally, Section 5.5 concludes the findings and examines possible directions for future work.

5.2 Related Work

5.2.1 FCE

FCE is widely used to assess injured workers before, during and after rehabilitation. A number of studies have demonstrated the reliability and validity of FCE and correlation with future recovery and return to work. Peppers et al. showed that augmenting clinical evaluation with FCE improves physicians' assessments of the patient's skills and work capacities [115]. Gross et al. studied the impact and benefits of integrating FCE into rehabilitation for better outcomes for injured workers [116]. FCE has been found to significantly predict return to work [117] and is an integral component of graded activity and functional rehabilitation programs [118]. However, James et al. concluded that further research is needed in FCE, especially on the use of computer technology (including robotics and digital sensors) [119].

5.2.2 Robot-assisted Assessment Rehabilitation

The inclusion of robots in therapy is becoming more common thanks to robots' power, repetitive motion ability, reprogramming capacity and potential adaptability to new tasks. These features allow robots to be used in therapy fields such as emotional therapy and physical therapy. Yakub et al. provide a list of robots developed in the context of rehabilitation medicine [120]. The use of robots in occupational rehabilitation began in the early 1990s [121][122], although they were employed mainly as assistive devices for workers with injury or disability. Recent developments in the area have culminated in devices such as BTE's EvalTech [113] and Simwork's ErgosII [114] systems, which simulate FCE assessment setups and can also be used for strength and movement coordination training. However, these devices are specifically designed with the emulation of a certain set of FCE tasks in mind. Also, the performance of tasks with these systems are spatially constrained to their placement on the devices and the performance of tasks involving free-space motions is not an option. For instance, while a device may include a lock for practicing turning a key to open it, the more challenging task for painting a wall is not supported because it cannot be done at one point on the device. The tasks also remain limited by the need to have physical objects that the user holds during assessments (e.g., rotating handles and knobs).

5.2.3 Virtual Reality & Augmented Reality in Rehabilitation

The virtual reality (VR) and augmented reality technology has been making its way into the rehabilitation field in recent years. It has been shown to increase the motivation of patients and keep them engaged since it uses games to disguise the repetitive movements of the rehabilitation exercises [67]. However, most of the VR and AR rehabilitation systems in the literature and on the market are targeted for those who have been affected by neurological injuries due to events such as stroke and spinal cord injury [123]. These systems cannot be used by injured workers as-is due to the difference in challenge level and sophistication of the rehabilitation tasks between the two groups (i.e., stroke patients and injured workers).

For non-immersive VR, in which the game is displayed in a 2D screen in front of the patient, there exist systems like the BTE Eccentron [124] to improve lower-limb strength while providing an interactive game-like experience to guide the patient toward their objectives. To the best of the authors' knowledge, there are currently no immersive VR or AR systems that train injured workers to regain muscle strength to enable them to return to work. There is also no robotic system that is specifically developed for simulating the physical dynamics of functional tasks for the rehabilitation of injured workers. Our proposed system employs the use of a 3D spatial AR display to immerse the patient in a projected 3D virtual environment that is integrated with the physical environment including the robotic manipulator. In the previous chapter, we showed that the resultant colocation of visual and motor axes help improve user performance in rehabilitation exercises [125].

We propose an approach based on using a 7 Degree-of-Freedom (DOF) serial manipulator for simulating the physical dynamics (i.e., haptic interaction) corresponding to functional tasks, eliminating the need for physical hardware of such tasks. Compared to rehabilitation facilities that allocate a large area for multiple tasks, this unified system can reduce the costs for equipment. Our approach also integrates an AR display to provide reconstructed visual feedback of the simulated task in an immersive environment. All types of motions can be performed on the robot due to its 7-DOF design. This allows flexibility in movement that is not found in other systems. Furthermore, having a robotic system allows for masking the task parameters from the patient which can help prevent the loss of motivation from knowing about an increase in the difficulty level of the task.

The overall robot-AR system is useful for both FCE and rehabilitation of injured workers. Serial manipulators have been previously incorporated into rehabilitation medicine for both assessment and rehabilitation purposes [49]. Our group, in particular, has extensively applied serial rehabilitation robots to target the neuromuscular rehabilitation of patients with stroke and cerebral palsy [126][127]. Likewise, we have also developed a robot-assisted AR system [125] for simulated stroke patients, in which the effects of stroke (e.g., being distracted) is simulated by cognitively loading the user with a count down task. However, to the best of our knowledge, the use of robots and AR in the context of facilitating FCE and rehabilitation of injured workers remains unexplored.

5.3 Materials and Methods

5.3.1 Rehabilitation Task Design

The simplified movements found in rehabilitation tasks often involve reaching, grasping, and weight lifting. The task used in our robot-AR system implements these movements in their basic forms but can be further adapted to higher difficulty and complexity levels.

We chose a painting task that focuses on training up-down hand movements by having the user paint a virtual wall. There is a fill indicator present to provide the user with information on the percentage of the wall they have already painted. Force feedback is also provided by the robotic manipulator when the virtual paint roller is in contact with the wall so that the haptic experience of painting on the wall in the real world is recreated. In the real-world condition, the user is given a physical paint roller to use on a portable physical wall positioned at the same spot the virtual wall was in the robot-AR condition. There is no paint used in the real case; rather, the user is asked to "paint over" an area of the wall as much as they can. The user continues "painting" until the area they have covered encompasses the wall in the virtual task.

Other measurements such as time of completion and amount of force exerted by the user can potentially be retrieved and analyzed for tasks in the robot-AR setup. However, since the metrics being evaluated in this work are the user's arm biomechanics, these measurements are not analyzed.

5.3.2 Robotic Manipulator Choice and Control Strategy

Many standardized FCEs such as the WorkWell FCE and the Progressive Isoinertial Lifting Evaluation (PILE) place emphasis on an injured worker's ability to lift weighted objects (e.g., crates) as an important assessment, among other physically strenuous tasks [111]. Therefore, it is desirable to use a robot capable of exerting enough force to realistically simulate heavy objects and interactions with environments typical of an injured worker's workplace. For this reason, the robot used in this work is a heavy-duty industrial robot; details are provided in Section 5.3.3. Internal gearing makes the structure of the robot non-back-drivable; however, the requirement of physical human-robot interaction (PHRI) in our experiments means a suitable robotic controller is needed to make the robot back-drivable.

Impedance controllers, which output a force for a robot to exert based on its motion, are ideal for providing stable PHRI when simulating environmental interactions.

However, implementing such controllers typically requires full knowledge of the robot's dynamics parameters such as each joint's mass and center of mass [128], which are unavailable for the robot used in this work and difficult to accurately measure. Admittance controllers, which output a motion for the robot to execute based on a measured force input, are a common alternative for non-back-drivable, heavy-duty robots like the one used in this system. The general form of an admittance controller's transfer function is

$$G = \frac{\vec{V}_d\left(s\right)}{\vec{W}\left(s\right)} = \frac{1}{Ms+B} \tag{5.1}$$

Note that in this system, an internal velocity controller is used by the robot to perform movements in real-time, so the admittance controller is designed here to output a desired velocity rather than a desired position. The input to the controller, $\vec{W}\left(s\right) = \left[\vec{F}\left(s\right), \vec{\tau}\left(s\right)\right]^\intercal$, represents the wrench composed of input forces and torques, and the output, $\vec{V}_d\left(s\right)$, is the resulting desired velocity, composed of Cartesian and angular terms. M represents the desired mass and inertia matrix and B represents the desired Cartesian and angular damping matrix. These matrices affect the transparency of free motion experienced by the user and also the stability of the robot. A stiffness parameter is not used, similar to [129], because restoring forces are not desirable during co-manipulation in free-space. It follows that G is given as

$$G = \begin{bmatrix} g_x & 0 & 0 & 0 & 0 & 0 \\ 0 & g_y & 0 & 0 & 0 & 0 \\ 0 & 0 & g_z & 0 & 0 & 0 \\ 0 & 0 & 0 & g_\alpha & 0 & 0 \\ 0 & 0 & 0 & 0 & g_\beta & 0 \\ 0 & 0 & 0 & 0 & 0 & g_\gamma \end{bmatrix}$$

where $\{g_x, g_y, \ldots, g_\gamma\}$ represent the admittance terms for each Cartesian direction and orientation angle. Large values for admittance terms result in greater allowed motions while small values result in more constrained movements. By changing the admittance

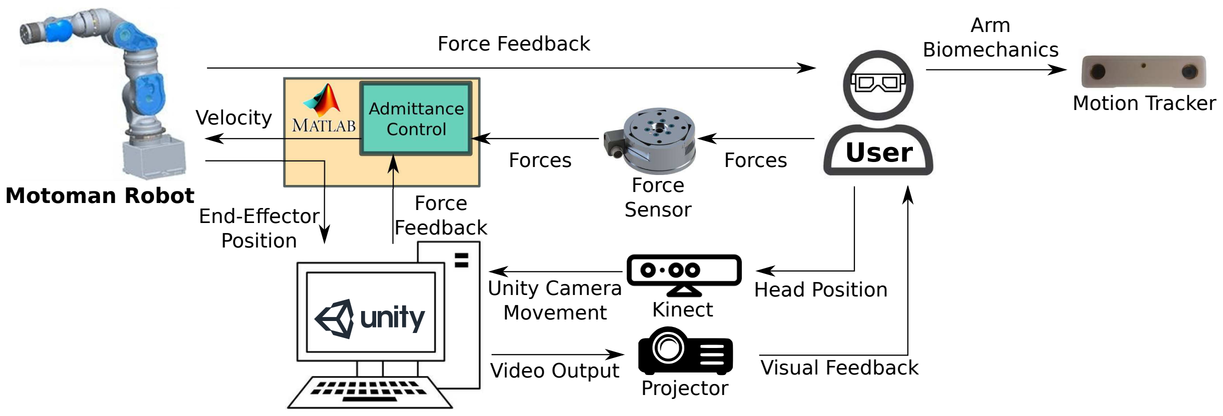


Figure 5.1: Flowchart of the communication between each system.

parameters, allowed movements initiated by the user can be restricted to certain axes. This is used, for example, in our tasks where it is beneficial to restrict rotations in axes that are not of interest (e.g., small values for g_β and g_γ) while allowing free motion in the other axes (e.g., large values for g_x, g_y, g_z, and g_α).

5.3.3 Experimental Setup

As seen in Fig.5.1, the robot-AR system uses a Motoman SIA-5F (Yaskawa America, Inc., Miamisburg, Ohio, USA) 7-DOF serial manipulator as the user interface to control the paint roller in the virtual environment. It is controlled using MATLAB, Simulink, and C++ in which the flow of communication between them is described in [130]. Attached to the robot's wrist joint before the end-effector is a 6-DOF ATI Gamma Net force/torque sensor (ATI Industrial Automation, Inc., Apex, North Carolina, USA). The AR subsystem consists of an off-the-shelf InFocus IN116A projector mounted 3 m above the ground that projects to a screen on the table. A Microsoft Kinect V2 Sensor is positioned 1.2 m horizontally distant and 0.34 m vertically above the user's head to enable head tracking for displaying the correct perspective to the user. To properly view the 3D scene, active DLP-Link 3D shutter glasses are worn by the user. The development of the 3D environment is done using the Unity Game Engine [91] where a virtual model of the workspace is created. This virtual model is created and calibrated to the world scale using Microsoft's RoomAlive Toolkit [106]. A ClaroNav MicronTracker (ClaroNav, Inc., Toronto, Ontario, Canada) motion tracking camera (MTC) is used to record the positions of the user's hand, elbow, and shoulder.

As mentioned earlier, the requirements of the more strenuous FCE and rehabilitation tasks imply a need for high force and torque haptic interactions. The admittance-type Motoman robot is used as the manipulator due to its heavy load capabilities compared to other impedance-type haptic interfaces [131]. It has a payload limit of 5 kg for accurate movement and can generate joint torques up to a rated 300 Nm. These are the maximum values achievable by the robot and we do not use all of it. For safety, constraints are placed in the software to limit high velocities and position singularities. The attached force sensor records user force and torque inputs, which are used to facilitate the admittance control of the robot.

The painting task requires a specific setup of the projector around the robot due to the limited projection space and required robot configuration. The configuration uses a curved screen with dimensions 85 cm tall, 75 cm deep, and 56 cm wide situated on top of the table. The end-effector of the robot is positioned to the right of the screen. This configuration allows the user to have an intuitive feel of the simulated task and reduces the occlusions on the projection display caused by the arm and joints of the user and the robot, respectively.

5.3.4 Experimental Procedure

5 trials for each condition (i.e., robot-AR or real-world) are carried out to have a total of 10 trials per person, lasting approximately 60 s per trial. The trials are performed by 2 able-bodied participants (both are male, 24 years old, and right-handed). Each participant is asked to stand in a comfortable position in front of the screen and to hold the Motoman robot's end-effector with his arm half extended. The participant is instructed to refrain from changing his standing location, which is marked on the floor, between the two experimental conditions. A chair is provided for the participant

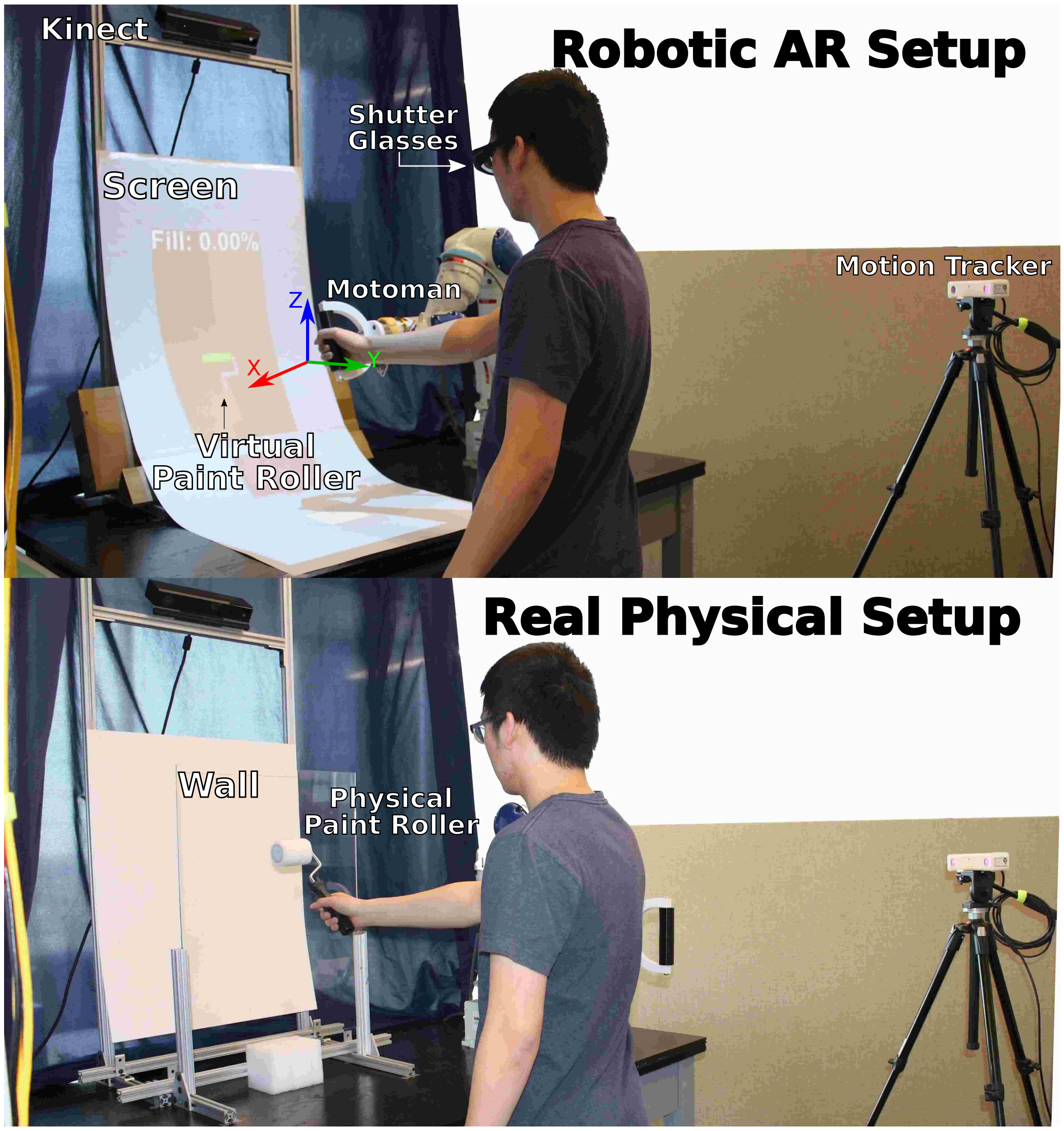


Figure 5.2: Painting task experimental setup. (Top) represents the robot-AR condition and the (Bottom) represents the real-life equivalent condition. The projector is not shown. Through AR, the paint roller will pop out in 3D from the perspective of the user in a geometrically correct position and orientation relative to the robot end-effector.

to take rests when needed. All 5 trials are recorded for a specific condition before moving onto the other one. The robot-AR condition for the painting task is presented to *Participant 1* as the first set of trials before doing the trials under the real-world condition. The opposite order is presented to *Participant 2*.

5.4 Results and Discussion

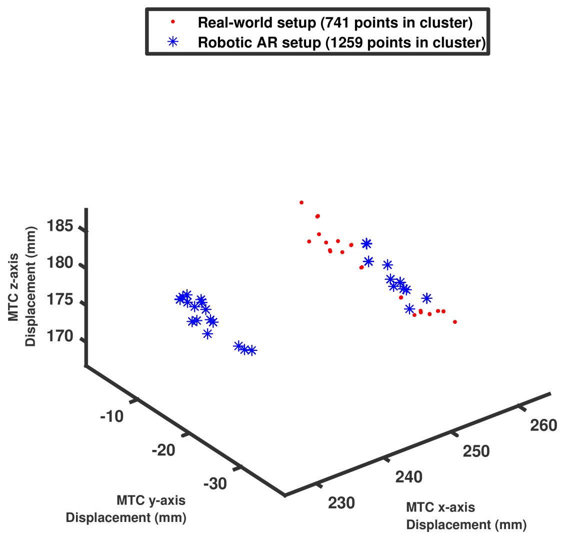
The hand, elbow, and shoulder positions recorded by the MTC form the data for the biomechanics analysis performed. These are recorded by putting fiducial markers behind the user's hand, on the elbow, and on the shoulder. In order to evaluate the similarity of the biomechanics between using the proposed system and the equivalent real-world tasks, we consider the hand position as the independent variable and the elbow and shoulder positions as the dependent variables. In other words, while the user's hand position changes, the elbow and shoulder joint positions will change in order to best accommodate the desired hand pose. For a fixed hand position (independent variable), we will compare the distribution of the dependent variables between the two conditions.

The two-sample Kolmogorov-Smirnov (KS) test is a well tested and commonly used method of evaluating whether two one-dimensional distributions are statistically different (the null hypothesis is that they are similar). Since p_{H-E} and p_{E-S} are in three dimensions, we use the modified version of the KS test in three dimensions as described in [132] by Fasano and Franceschini. We make use of the implementation of Fasano and Franceschini's work in [133] in conjunction with the Monte-Carlo simulations provided in the original work.

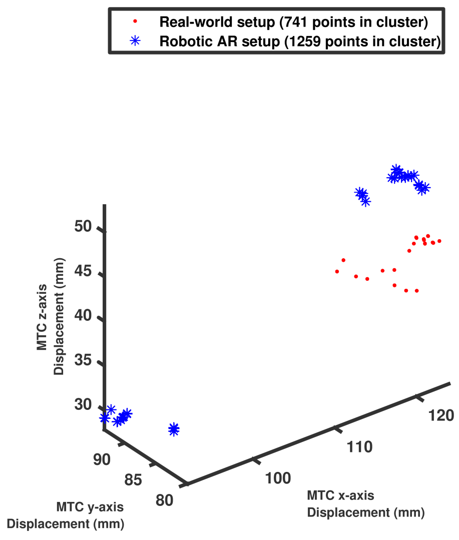
A preliminary comparison between the datasets for the two conditions is performed first to see if the distribution of elbow and shoulder joint positions for the same hand position as it traverses the entire surface of the wall being painted is statistically similar between the two conditions. Here, the joint position data from the real-world condition is taken as the baseline data; for a specific hand position, the elbow and shoulder positions for the robot-AR condition should resemble those measured in the real-world condition. If this happens, it can be concluded that the robot-AR system does not significantly modify how users perform the task compared to the real-world condition. The process of comparison is given as follows: for each recorded hand position in the robot-AR dataset, a similar hand position in the real-world data is found by using a nearest-neighbor (NN) search. For these similar hand positions, the associated hand-to-elbow (H2E) and elbow-to-shoulder (E2S) displacements can be calculated in each dataset. The result is a distribution of H2E and E2S displacements recorded in the robot-AR condition, and a distribution of H2E and E2S displacements that are associated with the real-world condition for the same hand positions. The modified KS test is then used to compare the H2E displacement distributions, and the E2S displacement distributions between the two conditions. Note that [132] only provides Monte Carlo simulations up to a maximum $n = 500$, where in a two sample KS test $n = \frac{n_1 n_2}{n_1 + n_2}$ where n_1 and n_2 are the number of points in the real-world condition and robot-AR condition, respectively. Knowing that the number of points in the distributions is the same, i.e., $n_1 = n_2$, we then restrict the number of points in the distributions to 1000 points or less. To do this, the collection of datapoints are downsampled to 1000 points for each condition, resulting in 200 points per trial. A significance value is returned by the test and is compared against an alpha value of $\alpha = 0.05$.

The results for both the H2E and E2S comparisons produced a value of $p < 0.05$, indicating that the distributions are statistically different (i.e., rejecting the null hypothesis that two conditions have the same distribution) and therefore suggests that the biomechanics of the two conditions differ in our painting task. This motivates a closer inspection of the data.

Examining whether there are spatial trends in the similarity between the distributions may help explain the dissimilarity reported in the KS test for the full dataset. To do this, we propose to divide the data down into spatial sections or voxels and perform the KS test for each individual voxel, looking for any that may be dissimilar. A grid of measurement points is first constructed by choosing points at evenly-spaced



(a)



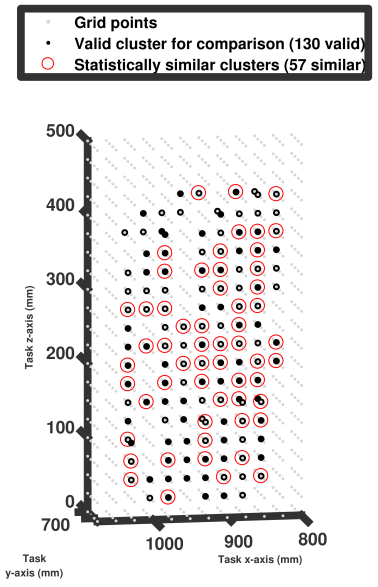
(b)

Figure 5.3: Joint position data for an example cluster. (a) shows the point cloud data for H2E displacements, and (b) shows E2S displacements.

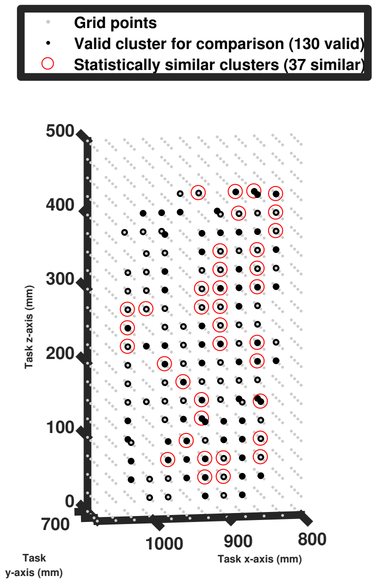
intervals to encompass the range of hand positions across all datasets in the three Cartesian dimensions. All recorded hand positions are then clustered to the nearest grid point using the NN search. We use an interval of 25 *mm* as the distance between grid points, as it provides a high resolution of voxels in our task space and allows most clusters to meet the requirements for n_1 and n_2. Fig. 5.3 shows the distributions of H2E displacements and E2S displacements for an example cluster.

For each cluster, the statistical similarity of the distributions for the associated data from the two conditions (robot-AR and real-world setups) is then evaluated with the modified KS test. Similar to before, the Monte Carlo simulations in [132] are only provided for a minimum of $n = 10$ between two samples (and a maximum of $n = 500$). To ensure $10 \leq n \leq 500$, we impose limits where $n_1 \geq 20, n_2 \geq 20$ and $n_1 + n_2 \leq 2000$. Graphical results of the modified KS tests are shown in Fig. 5.4.

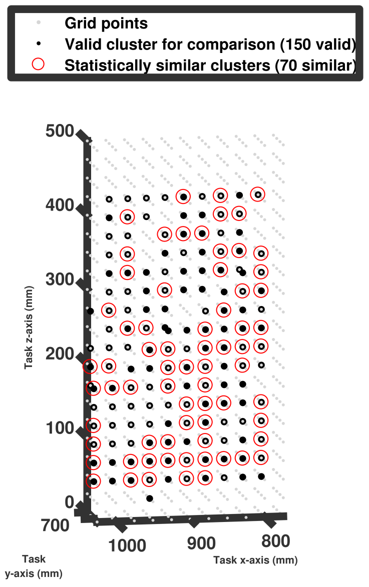
The percentage of clusters that are statistically similar between the two conditions show that 43.85% of the measurable clusters were similar for H2E displacements and 28.46% were similar for E2S displacements during Participant 1's trials. Participant 2



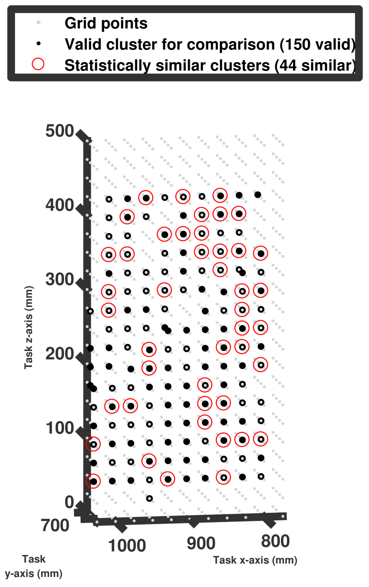
(a)



(b)



(c)



(d)

Figure 5.4: Three-dimensional KS results for the painting task with the data split into voxels for comparison. The grid points show points in space around the surface of the wall where H2E and E2S results were clustered and compared at. (a) and (b) represent H2E and E2S results for *Participant 1*, respectively, and (c) and (d) represent the same for *Participant 2*. Clusters with a sufficient number of datapoints for comparison with the KS test are shown with solid black points and those of statistical similarity are encircled in red.

achieved similarity with 46.67% similar for H2E displacements and 29.33% similarity for E2S displacements.

At first glance, the fraction of clusters that produced similar results seems to be quite low, especially for the E2S displacements. However, a qualitative observation of the results in Fig. 5.4 shows that the statistically similar clusters are well spread across the entire workspace. There are a few possible reasons as to why some clusters may not show similar results. As the real-world condition experiments did not involve actual paint being laid on the physical wall, keeping track of the "painted" portion proved to be challenging. This could affect the fairness of the KS test performed. For example, if, for a specific cluster, $n_1 \gg n_2$, where $n_1, n_2 > 20$ and $n_1 + n_2 \leq 2000$, then a comparison of the distributions would be valid according to the restrictions we placed on the comparison in a cluster, but it could suffer from the disparity in the quality of the distributions. The simplest way to address this issue would be to simply have more trials, which in turn would provide more data and a higher chance to better define the distributions for more clusters. It would also be highly beneficial in this situation to be able to remove the upper limit on datapoints to compare over, meaning running Monte Carlo simulations as in [132] for higher values of n.

Nevertheless, the results indicate that there is a perceivable difference between using the robot-AR setup and performing the real-world equivalent task. The most likely cause would be that the damping and inertia of the robot were not low enough to properly convey full transparency during free motion. This could be the case, given the nature of the geared transmission system used in the robot and the admittance controller used to make it compliant. The implied difference in perceived weight during free motion would then be a likely cause in any changes in the observed biomechanics, as the user would compensate for the heavier load, experienced in the robot-AR condition, by adjusting their joint positions accordingly. In the results shown in Fig. 5.4, this may be the reason why the E2S distributions have a much lower overall similarity than the H2E distributions, as the upper arm may have moved more in order to compensate for the larger resistance to motion in the robot-AR condition while the lower arm remained the same in order to hold the brush handle comfortably. There is then a motivation for reexamining the results when performed using robot with similar load-bearing capabilities that is designed for the purpose of patient-safe interaction as well as built-in back-drivability, such as the Franka Emika Panda 7-DOF robot (Franka Emika GmbH, Munich, Germany).

5.5 Conclusion

In this chapter, a robot-AR system that aims to be a suitable alternative to existing FCE and rehabilitation environments for injured or disabled workers is developed and evaluated. A task that involves painting a wall is presented to the participants. To evaluate our approach, the task has a real-life equivalent condition in which the biomechanics of the participant's arm for both robotic AR and real-life conditions are recorded and compared to determine if the arm movements are similar. Our results show that the arm biomechanics for a painting task are statistically similar in $\approx 50\%$ of the collected clusters for the hand-to-elbow (H2E) displacements, and $\approx 30\%$ for the elbow-to-shoulder (E2S) displacements for both participants. These initial findings show the potential of our robot-AR system to replicate upper-limb movements found in traditional FCE and rehabilitation exercises and it motivates us to further investigate better methods to simulate functional tasks. Future work includes implementing the system with a different robot and/or robot controller that is better suited for physical human-robot interaction (PHRI), expanding the projected area, developing more tasks, and testing the system with actual FCE tasks, or even in a clinical setting. By creating an all-in-one robotic AR occupational rehabilitation system, we hope to motivate and provide an efficient method for workers to recover from their injuries.

Chapter 6

Conclusion

6.1 Summary

This thesis presented an investigation of the improvement of user performance in rehabilitation exercise environments in which the user's motor and visual frames are colocated. Through an implementation of an augmented-reality (AR) display, it is possible to align the movements of the patient's arm with the movement of their avatar or cursor on the screen in terms of location, direction, and scale. The aim of this work was to make rehabilitation easier for patients who have had their mental capabilities affected negatively due to disabling events such as stroke. Since AR is currently an emerging field of technology, the current literature that applies it to a rehabilitation context is not extensive. This work introduced the first steps toward this direction and showed the potential for this new technology to be integrated in the rehabilitation field.

In Chapter 3, we began to examine the effects on user performance of AR integration in experiments involving a 2-DOF planar rehabilitation robot. AR was strictly implemented in a 2D environment and the participants were instructed to perform a rehabilitation reaching exercise in the form a game projected onto the same table that the robot is placed on. In this experiment, the participants pushed a virtual projection of a 2D car using the end-effector of the robot along a circular track around the workspace. The experiment compared the user performance, with respect to time, of the participants in eight different conditions involving the presence or absence of three independent factors: visual colocation, haptic feedback, and cognitive loading. A brief analysis of the participants' spatial error between the desired trajectory and actual trajectory was showcased in a snapshot of the data of two participants. A survey was also given after the experiments to get feedback on what the participants thought about the system.

In Chapter 4, instead of a 2D environment, we further investigated the matter in a 3D AR system. A projection setup with a curved screen was employed, and head tracking with a Kinect was utilized. A 6-DOF haptic device was used by the participants for each of the three games that were created to mimic rehabilitation exercises: snapping, catching, and ball dropping. The snapping task required careful and precise maneuvers to traverse past the floating obstacles and reach the highlighted point. The catching task tested the participants' reaction time and speed in hand movement to catch the falling spheres. Finally, the ball dropping task needed accurate positioning of the ball in order for it to fall into the hole. Each task was analyzed separately; however, four conditions were tested for each one. The presence and absence of colocation and cognitive loading were the independent factors of each condition. Haptic feedback was always present for consistency due to having different levels of involvement in each task. User performance was based on the success scores achieved by participants depending on the task. The snapping task results showed an unintended effect of cognitive loading; the participants moved more carefully and therefore made fewer mistakes than when cognitive loading was not present. For both catching and ball dropping, an improvement in user performance was seen as hypothesized.

Chapter 5 brought together the results and observations taken from the previous chapters to create a robotic system with an AR display for functional capacity evaluation (FCE) and rehabilitation of injured workers. We presented a system comprised of a robotic arm for recreating the physical dynamics of functional tasks and a 3D AR display for immersive visualization of the tasks. We chose a painting task that focused on training up-down hand movements by having the user paint a virtual wall. Participants performed a virtual version of the task using the robot-AR system, and a physical version of the same task without the system. This study showed the results for two able-bodied users, each analyzed separately, to determine if the robot-AR system produces upper-limb movements similar to the real-life equivalent task. The similarity between relative joint positions, i.e., hand-to-elbow (H2E) and elbow-to-shoulder (E2S) displacements, was evaluated within clusters based on the spatial position of the user's hand. The H2E displacements for approximately 50% of hand position clusters were consistent between the robot-AR and real-world conditions and approximately 30% for E2S displacements. The similar clusters were distributed across the entire task space however, indicating the robot-AR system has the potential to properly simulate a real-world equivalent task.

6.2 Future Work

6.2.1 Assist-as-needed Functionality

For patients who are unable to fully complete the tasks due to limb tremors or complications such as the inability to fully extend their arm, incorporating an assist-as-needed functionality with the haptic device would be ideal. The tasks can be programmed to help the patient by gravitating the end-effector towards the desired position. However, the robot moving the patient's arm will have no positive effect unless the patient attempts to do the task. Thus, the robot should only help a patient who is putting his/her maximum effort in. This may be achieved by detecting the effort produced by the patient's arm through surface electromyography (sEMG), or an electroencephalogram (EEG) to measure their mental activity and then providing assistance depending on a certain threshold or control scheme. Simpler methods may be implementing a position controller with a variable gain towards the direction of the desired position to ensure the patient is constantly moving towards the target. Implementation of this functionality may help increase success rates and further motivate patients through their rehabilitation process.

6.2.2 Larger Scale Setup

The work can be extended to a larger AR environment similar to the CAVE systems. This would involve the addition of multiple projectors and depth sensors, and ensuring seamless colour compensation for overlapping projections. A larger system would open up the potential for more versatile rehabilitation tasks and accommodate different types of patients (i.e., allowing movement on a wheelchair). For example, a person on a wheelchair can move to an area of the room with a projected sink. The tap handle can be represented by a robot end-effector, allowing the patient to open the tap and see water flow in the AR display.

6.2.3 Improved Occlusion Mitigation

Since occlusion is present in the system, the task had to be designed in a way that it minimizes its effects. Occlusion happens when an object (i.e., the user's hand or the robot) gets in the way of the projection space where the virtual object is meant to be projected on. The object occludes the projection and casts a shadow on the screen instead which can cause a break in immersion in an AR setup. However, proper configuration of the depth sensors and the addition of more projectors should allow for the detection of real objects onto the scene and reduce the effect of shadows such that the virtual environment can respond dynamically to the added object and project the correct perspective for the user.

6.2.4 Therapist-Patient Telerehabilitation

If an implementation similar to Benko's work with the mirage table and its 3D teleconferencing interface [105] is achieved, then patients might be more motivated to do home-based rehabilitation. Both the patient and the therapist can interact within the shared interface as if they sat at a table across from each other.

6.2.5 Clinical Trials and Validation

The main goal of this research is to improve the user performance during rehabilitation exercises to provide more effective rehabilitation outcomes. Therefore, it is crucial to validate the system by conducting a longitudinal study on actual patients that have a disability. The system can be compared with a similar traditional rehabilitation setup by analyzing the outcomes to determine its effectiveness against current methods.

Bibliography

[1] "Stroke report 2016." http://www.strokebestpractices.ca/news-feature/stroke-report-2016-just-released/.

[2] N. Maclean, P. Pound, C. Wolfe, and A. Rudd, "A critical review of the concept of patient motivation in the literature on physical rehabilitation," *Soc Sci Med*, vol. 50, no. 4, pp. 495–506, 2000.

[3] R. Colombo, F. Pisano, A. Mazzone, C. Delconte, S. Micera, M. C. Carrozza, P. Dario, and G. Minuco, "Design strategies to improve patient motivation during robot-aided rehabilitation," *Journal of neuroengineering and rehabilitation*, vol. 4, no. 1, p. 3, 2007.

[4] T. M. Damush, L. Plue, T. Bakas, A. Schmid, and L. S. Williams, "Barriers and facilitators to exercise among stroke survivors," *Rehabilitation nursing*, vol. 32, no. 6, pp. 253–262, 2007.

[5] M. Ma, M. McNeill, D. Charles, S. McDonough, J. Crosbie, L. Oliver, and C. McGoldrick, "Adaptive virtual reality games for rehabilitation of motor disorders," in *International Conference on Universal Access in Human-Computer Interaction*, pp. 681–690, Springer, 2007.

[6] A. L. Betker, A. Desai, C. Nett, N. Kapadia, and T. Szturm, "Game-based exercises for dynamic short-sitting balance rehabilitation of people with chronic spinal

cord and traumatic brain injuries," *Physical therapy*, vol. 87, no. 10, pp. 1389–1398, 2007.

[7] J. R. Octavia and K. Coninx, "Adaptive personalized training games for individual and collaborative rehabilitation of people with multiple sclerosis," *BioMed research international*, vol. 2014, 2014.

[8] K. Brütsch, A. Koenig, L. Zimmerli, S. Mérillat-Koeneke, R. Riener, L. Jäncke, H. J. van Hedel, and A. Meyer-Heim, "Virtual reality for enhancement of robot-assisted gait training in children with neurological gait disorders," *Journal of rehabilitation medicine*, vol. 43, no. 6, pp. 493–499, 2011.

[9] A. Mirelman, P. Bonato, and J. E. Deutsch, "Effects of training with a robot-virtual reality system compared with a robot alone on the gait of individuals after stroke," *Stroke*, vol. 40, no. 1, pp. 169–174, 2009.

[10] M. Hillman, "2 rehabilitation robotics from past to present–a historical perspective," in *Advances in Rehabilitation Robotics*, pp. 25–44, Springer, 2004.

[11] D. Khalili and M. Zomlefer, "An intelligent robotic system for rehabilitation of joints and estimation of body segment parameters," *IEEE transactions on biomedical engineering*, vol. 35, no. 2, pp. 138–146, 1988.

[12] N. Hogan, H. I. Krebs, J. Charnnarong, P. Srikrishna, and A. Sharon, "Mit-manus: a workstation for manual therapy and training. i," in *Robot and Human Communication, 1992. Proceedings., IEEE International Workshop on*, pp. 161–165, IEEE, 1992.

[13] P. S. Lum, C. G. Burgar, and P. C. Shor, "Evidence for improved muscle activation patterns after retraining of reaching movements with the mime robotic system in subjects with post-stroke hemiparesis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 12, no. 2, pp. 186–194, 2004.

[14] D. J. Reinkensmeyer, L. E. Kahn, M. Averbuch, A. McKenna-Cole, B. D. Schmit, and W. Z. Rymer, "Understanding and treating arm movement impairment after chronic brain injury: progress with the arm guide.," 2014.

[15] D. J. Williams, H. I. Krebs, and N. Hogan, "A robot for wrist rehabilitation," in *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, vol. 2, pp. 1336–1339, IEEE, 2001.

[16] T. Worsnopp, M. Peshkin, J. Colgate, and D. Kamper, "An actuated finger exoskeleton for hand rehabilitation following stroke," in *Rehabilitation Robotics, 2007. ICORR 2007. IEEE 10th International Conference on*, pp. 896–901, IEEE, 2007.

[17] G. Colombo, M. Joerg, R. Schreier, V. Dietz, *et al.*, "Treadmill training of paraplegic patients using a robotic orthosis," *Journal of rehabilitation research and development*, vol. 37, no. 6, pp. 693–700, 2000.

[18] J. E. Deutsch, J. Latonio, G. C. Burdea, and R. Boian, "Post-stroke rehabilitation with the rutgers ankle system: a case study," *Presence: Teleoperators & Virtual Environments*, vol. 10, no. 4, pp. 416–430, 2001.

[19] B. Volpe, H. Krebs, N. Hogan, L. Edelsteinn, C. Diels, and M. Aisen, "Robot training enhanced motor outcome in patients with stroke maintained over 3 years," *Neurology*, vol. 53, no. 8, pp. 1874–1874, 1999.

[20] P. S. Lum, C. G. Burgar, P. C. Shor, M. Majmundar, and M. Van der Loos, "Robot-assisted movement training compared with conventional therapy techniques for the rehabilitation of upper-limb motor function after stroke," *Archives of physical medicine and rehabilitation*, vol. 83, no. 7, pp. 952–959, 2002.

[21] G. Alankus, R. Proffitt, C. Kelleher, and J. Engsberg, "Stroke therapy through motion-based games: a case study," *ACM Transactions on Accessible Computing (TACCESS)*, vol. 4, no. 1, p. 3, 2011.

[22] C. C. Abt, *Serious games.* University press of America, 1987.

[23] B. Sawyer and D. Rejeski, “Serious games: Improving public policy through game-based learning and simulation,” 2002.

[24] J. Alvarez, V. Alvarez, D. Djaouti, and L. Michaud, “Serious games: Training & teaching-healthcare-defence & security-information & communication,” *IDATE, France*, 2010.

[25] A. D’Angour, “Plato and play: Taking education seriously in ancient greece.,” *American Journal of Play*, vol. 5, no. 3, pp. 293–307, 2013.

[26] “The Single Tax Review,” Autumn, 1902. http://landlordsgame.info/articles/LLG_SingleTaxReview-1902.pdf.

[27] “How Henry George’s Principles Were Corrupted Into the Game Called Monopoly,” Dec 2011. http://www.henrygeorge.org/dodson_on_monopoly.htm.

[28] J. Rice, “Assessing higher order thinking in video games,” *Journal of Technology and Teacher Education*, vol. 15, no. 1, pp. 87–100, 2007.

[29] “A Pioneering Game’s Journey: The History of Oregon Trail,” Apr 2017. https://www.usgamer.net/articles/the-oral-history-of-oregon-trail.

[30] W. Wright and I. Bogost, *Persuasive games: The expressive power of videogames.* Mit Press, 2007.

[31] R. Shilling, M. Zyda, and E. C. Wardynski, “Introducing emotion into military simulation and videogame design: America’s army operations and virte,” 2002.

[32] S. Göbel, S. Hardy, V. Wendel, F. Mehm, and R. Steinmetz, “Serious games for health: personalized exergames,” in *Proceedings of the 18th ACM international conference on Multimedia*, pp. 1663–1666, ACM, 2010.

[33] D. Thompson, T. Baranowski, R. Buday, J. Baranowski, V. Thompson, R. Jago, and M. J. Griffith, "Serious video games for health: How behavioral science guided the development of a serious video game," *Simulation & gaming*, vol. 41, no. 4, pp. 587–606, 2010.

[34] J. E. Deutsch, M. Borbely, J. Filler, K. Huhn, and P. Guarrera-Bowlby, "Use of a low-cost, commercially available gaming console (wii) for rehabilitation of an adolescent with cerebral palsy," *Physical therapy*, vol. 88, no. 10, pp. 1196–1207, 2008.

[35] S. Flynn, P. Palma, and A. Bender, "Feasibility of using the sony playstation 2 gaming platform for an individual poststroke: a case report," *Journal of neurologic physical therapy*, vol. 31, no. 4, pp. 180–189, 2007.

[36] A. Da Gama, T. Chaves, L. Figueiredo, and V. Teichrieb, "Poster: improving motor rehabilitation process through a natural interaction based system using kinect sensor," in *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pp. 145–146, IEEE, 2012.

[37] "Serious Play Conference." https://seriousplayconf.com/.

[38] B. Grahn, C. Ekdahl, and L. Borgquist, "Motivation as a predictor of changes in quality of life and working ability in multidisciplinary rehabilitation," *Disability and Rehabilitation*, vol. 22, no. 15, pp. 639–654, 2000.

[39] N. Maclean, P. Pound, C. Wolfe, and A. Rudd, "Qualitative analysis of stroke patients' motivation for rehabilitation," *Bmj*, vol. 321, no. 7268, pp. 1051–1054, 2000.

[40] K. Ekberg, "Workplace changes in successful rehabilitation," *Journal of Occupational Rehabilitation*, vol. 5, no. 4, pp. 253–269, 1995.

[41] E. J. Lenze, M. C. Munin, T. Quear, M. A. Dew, J. C. Rogers, A. E. Begley, and C. F. Reynolds III, "Significance of poor patient participation in physical

and occupational therapy for functional outcome and length of stay," *Archives of physical medicine and rehabilitation*, vol. 85, no. 10, pp. 1599–1601, 2004.

[42] J. M. Sietsema, D. L. Nelson, R. M. Mulder, D. Mervau-Scheidel, and B. E. White, "The use of a game to promote arm reach in persons with traumatic brain injury," *American Journal of Occupational Therapy*, vol. 47, no. 1, pp. 19–24, 1993.

[43] A. C. Mosey, *Psychosocial components of occupational therapy.* Lippincott Williams & Wilkins, 1986.

[44] D. J. Reinkensmeyer, C. T. Pang, J. A. Nessler, and C. C. Painter, "Web-based telerehabilitation for the upper extremity after stroke," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 10, no. 2, pp. 102–108, 2002.

[45] C. Jadhav and V. Krovi, "A low-cost framework for individualized interactive telerehabilitation," in *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*, vol. 2, pp. 3297–3300, IEEE, 2004.

[46] M. Sandlund, S. McDonough, and C. Häger-Ross, "Interactive computer play in rehabilitation of children with sensorimotor disorders: a systematic review," *Developmental Medicine & Child Neurology*, vol. 51, no. 3, pp. 173–179, 2009.

[47] D. Novak, A. Nagle, U. Keller, and R. Riener, "Increasing motivation in robot-aided arm rehabilitation with competitive and cooperative gameplay," *Journal of neuroengineering and rehabilitation*, vol. 11, no. 1, p. 64, 2014.

[48] M. Maier, B. R. Ballester, E. Duarte, A. Duff, and P. F. Verschure, "Social integration of stroke patients through the multiplayer rehabilitation gaming system," in *International Conference on Serious Games*, pp. 100–114, Springer, 2014.

[49] H. I. Krebs, N. Hogan, M. L. Aisen, and B. T. Volpe, "Robot-aided neurorehabilitation," *IEEE transactions on rehabilitation engineering*, vol. 6, no. 1, pp. 75–87, 1998.

[50] D. Djaouti, J. Alvarez, and J.-P. Jessel, "Classifying serious games: the g/p/s model," in *Handbook of research on improving learning and motivation through educational games: Multidisciplinary approaches*, pp. 118–136, IGI Global, 2011.

[51] H. G. Hoffman, D. R. Patterson, and G. J. Carrougher, "Use of virtual reality for adjunctive treatment of adult burn pain during physical therapy: a controlled study," *The Clinical journal of pain*, vol. 16, no. 3, pp. 244–250, 2000.

[52] H. Pardini, "VR Vaccine." https://www.adforum.com/award-organization/6650183/showcase/2017/ad/34544861.

[53] E. Steele, K. Grimmer, B. Thomas, B. Mulley, I. Fulton, and H. Hoffman, "Virtual reality as a pediatric pain modulation technique: a case study," *Cyberpsychology & Behavior*, vol. 6, no. 6, pp. 633–638, 2003.

[54] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *IEICE TRANSACTIONS on Information and Systems*, vol. 77, no. 12, pp. 1321–1329, 1994.

[55] A. Garcia, N. Andre, D. Bell Boucher, A. Roberts-South, M. Jog, and M. Katchabaw, *Immersive Augmented Reality for Parkinson Disease Rehabilitation*, pp. 445–469. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[56] D. Van Krevelen and R. Poelman, "A survey of augmented reality technologies, applications and limitations," *International journal of virtual reality*, vol. 9, no. 2, p. 1, 2010.

[57] S. F. Atashzar, M. Naish, and R. V. Patel, "5 active sensorimotor augmentation in robotics-assisted surgical systems," in *Mixed and Augmented Reality in Medicine*, pp. 61–81, CRC Press, 2018.

[58] "ReJoyce by rehabtronics." https://www.blog.rehabtronics.com/rejoyce.

[59] S. Côté and S. Bouchard, "Virtual reality exposure for phobias: A critical review," *Journal of CyberTherapy & Rehabilitation*, vol. 1, no. 1, pp. 75–91, 2008.

[60] N. Robitaille, P. L. Jackson, L. J. Hébert, C. Mercier, L. J. Bouyer, S. Fecteau, C. L. Richards, and B. J. McFadyen, "A virtual reality avatar interaction (vrai) platform to assess residual executive dysfunction in active military personnel with previous mild traumatic brain injury: proof of concept," *Disability and Rehabilitation: Assistive Technology*, vol. 12, no. 7, pp. 758–764, 2017.

[61] D. L. Jaffe, D. A. Brown, C. D. Pierson-Carey, E. L. Buckley, and H. L. Lew, "Stepping over obstacles to improve walking in individuals with poststroke hemiplegia.," *Journal of Rehabilitation Research & Development*, vol. 41, 2004.

[62] L. Connelly, Y. Jia, M. L. Toro, M. E. Stoykov, R. V. Kenyon, and D. G. Kamper, "A pneumatic glove and immersive virtual reality environment for hand rehabilitative training after stroke," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 18, no. 5, pp. 551–559, 2010.

[63] X. Casas, G. Herrera, I. Coma, and M. Fernández, "A kinect-based augmented reality system for individuals with autism spectrum disorders.," in *Grapp/ivapp*, pp. 440–446, 2012.

[64] M. C. Juan and J. Calatrava, "An augmented reality system for the treatment of phobia to small animals viewed via an optical see-through hmd: comparison with a similar system viewed via a video see-through hmd," *International Journal of Human-Computer Interaction*, vol. 27, no. 5, pp. 436–449, 2011.

[65] J. Vieira, M. Sousa, A. Arsénio, and J. Jorge, "Augmented reality for rehabilitation using multimodal feedback," in *Proceedings of the 3rd 2015 Workshop on ICTs for improving Patients Rehabilitation Research Techniques*, pp. 38–41, ACM, 2015.

[66] M. Shaughnessy, B. M. Resnick, and R. F. Macko, “Testing a model of post-stroke exercise behavior,” *Rehabil Nurs*, vol. 31, pp. 15–21, 2006.

[67] D. J. Reinkensmeyer and S. J. Housman, “”If I can’t do it once, why do it a hundred times?”: Connecting volition to movement success in a virtual environment motivates people to exercise the arm after stroke,” in *2007 Virtual Rehabilitation*, pp. 44–48, Sept 2007.

[68] “Oculus rift.” https://www.oculus.com/.

[69] “Microsoft hololens.” https://www.microsoft.com/en-ca/hololens.

[70] A. Alamri, J. Cha, and A. E. Saddik, “Ar-rehab: An augmented reality framework for poststroke-patient rehabilitation,” *IEEE Transactions on Instrumentation and Measurement*, vol. 59, pp. 2554–2563, Oct 2010.

[71] H. Mousavi Hondori, M. Khademi, L. Dodakian, S. C. Cramer, and C. V. Lopes, “A Spatial Augmented Reality rehab system for post-stroke hand rehabilitation,” *Stud Health Technol Inform*, vol. 184, pp. 279–285, 2013.

[72] M. C. Juan and J. Calatrava, “An augmented reality system for the treatment of phobia to small animals viewed via an optical see-through hmd: Comparison with a similar system viewed via a video see-through hmd,” *International Journal of Human–Computer Interaction*, vol. 27, no. 5, pp. 436–449, 2011.

[73] “Kinova robotics.” http://www.kinovarobotics.com/.

[74] Y. Ikeda, E. Suzuki, T. Kuramata, T. Kozaki, T. Koyama, Y. Kato, Y. Murakami, H. Enaida, and T. Ishibashi, “Development and evaluation of a visual aid using see-through display for patients with retinitis pigmentosa,” *Japanese Journal of Ophthalmology*, vol. 59, pp. 43–47, Jan 2015.

[75] S. C. Yeh, W. Y. Hwang, T. C. Huang, W. K. Liu, Y. T. Chen, and Y. P. Hung, “A study for the application of body sensing in assisted rehabilitation training,”

in *2012 International Symposium on Computer, Consumer and Control*, pp. 922–925, June 2012.

[76] FGTeam, "Rejoyce speeds up upper extremity recovery post stroke," 2015. https://www.fitness-gaming.com/news/health-and-rehab/rejoyce-speeds-up-upper-extremity-recovery-post-stroke.html.

[77] C. Kaminer, K. LeBras, J. McCall, T. Phan, P. Naud, M. Teodorescu, and S. Kurniawan, "An immersive physical therapy game for stroke survivors," in *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility*, ASSETS '14, (New York, NY, USA), pp. 299–300, ACM, 2014.

[78] J. W. Burke, M. D. J. McNeill, D. K. Charles, P. J. Morrow, J. H. Crosbie, and S. M. McDonough, "Augmented reality games for upper-limb stroke rehabilitation," in *2010 Second International Conference on Games and Virtual Worlds for Serious Applications*, pp. 75–78, March 2010.

[79] A. G. D. Correa, G. A. de Assis, M. d. Nascimento, I. Ficheman, and R. d. D. Lopes, "Genvirtual: An augmented reality musical game for cognitive and motor rehabilitation," in *2007 Virtual Rehabilitation*, pp. 1–6, Sept 2007.

[80] J. Trojan, M. Diers, X. Fuchs, F. Bach, R. Bekrater-Bodmann, J. Foell, S. Kamping, M. Rance, H. Maass, and H. Flor, "An augmented reality home-training system based on the mirror training and imagery approach," *Behav Res Methods*, vol. 46, pp. 634–640, Sep 2014.

[81] S. V. Adamovich, A. S. Merians, R. Boian, M. Tremaine, G. S. Burdea, M. Recce, and H. Poizner, "A virtual reality based exercise system for hand rehabilitation post-stroke: transfer to function," in *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2, pp. 4936–4939, Sept 2004.

[82] V. H. Andaluz, P. J. Salazar, M. Escudero V., C. Bustamante D., M. Silva S., W. Quevedo, J. S. Sánchez, E. G. Espinosa, and D. Rivas, “Virtual reality integration with force feedback in upper limb rehabilitation,” in *Advances in Visual Computing*, (Cham), pp. 259–268, Springer International Publishing, 2016.

[83] C. Vidrios-Serrano, I. Bonilla, F. Vigueras-Gómez, and M. Mendoza, “Development of a haptic interface for motor rehabilitation therapy using augmented reality,” in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 1156–1159, Aug 2015.

[84] X. Luo, T. Kline, H. C. Fischer, K. A. Stubblefield, R. V. Kenyon, and D. G. Kamper, “Integration of augmented reality and assistive devices for post-stroke hand opening rehabilitation,” in *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pp. 6855–6858, 2005.

[85] M. Khademi, H. M. Hondori, C. V. Lopes, L. Dodakian, and S. C. Cramer, “Haptic augmented reality to monitor human arm’s stiffness in rehabilitation,” in *2012 IEEE-EMBS Conference on Biomedical Engineering and Sciences*, pp. 892–895, Dec 2012.

[86] M. Khademi, H. M. Hondori, L. Dodakian, S. Cramer, and C. V. Lopes, “Comparing ”pick and place” task in spatial Augmented Reality versus non-immersive Virtual Reality for rehabilitation setting,” *Conf Proc IEEE Eng Med Biol Soc*, vol. 2013, pp. 4613–4616, 2013.

[87] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, “Surround-screen projection-based virtual reality: the design and implementation of the cave,” in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 135–142, ACM, 1993.

[88] J. Westwood *et al.*, "Caren-computer assisted rehabilitation environment," *Medicine Meets Virtual Reality: The Convergence of Physical & Informational Technologies: Options for a New Era in Healthcare*, vol. 62, p. 373, 1999.

[89] A. Hussain, S. Balasubramanian, N. Roach, J. Klein, N. Jarrassé, M. Mace, A. David, S. Guy, and E. Burdet, "Sitar: a system for independent task-oriented assessment and rehabilitation," *Journal of Rehabilitation and Assistive Technologies Engineering*, vol. 4, p. 2055668317729637, 2017.

[90] A. A. Rizzo, D. Strickland, and S. Bouchard, "The challenge of using virtual reality in telerehabilitation," *Telemedicine Journal & E-Health*, vol. 10, no. 2, pp. 184–195, 2004.

[91] "Unity." https://unity3d.com/.

[92] M. Plotnik, Y. Dagan, T. Gurevich, N. Giladi, and J. M. Hausdorff, "Effects of cognitive function on gait and dual tasking abilities in patients with parkinson's disease suffering from motor response fluctuations," *Experimental Brain Research*, vol. 208, pp. 169–179, Jan 2011.

[93] M. Lezak, D. Howieson, and D. Loring, *Neuropsychological assessment.* Oxford University Press, New York, fourth ed., 2004.

[94] A. D. Wilson and H. Benko, "Combining multiple depth cameras and projectors for interactions on, above and between surfaces," in *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, (New York, NY, USA), pp. 273–282, ACM, 2010.

[95] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[96] "Spss statistics." https://www.ibm.com/analytics/spss-statistics-software.

[97] C. A. Marshall, “An analysis of motivation as a predictor of vocational rehabilitation outcomes,” 1989.

[98] L. S. Williams, S. S. Ghose, and R. W. Swindle, “Depression and other mental health diagnoses increase mortality risk after ischemic stroke,” *American Journal of Psychiatry*, vol. 161, no. 6, pp. 1090–1095, 2004.

[99] A. E. F. Da Gama, T. M. Chaves, L. S. Figueiredo, A. Baltar, M. Meng, N. Navab, V. Teichrieb, and P. Fallavollita, “Mirrarbilitation: A clinically-related gesture recognition interactive tool for an ar rehabilitation system,” *Computer methods and programs in biomedicine*, vol. 135, pp. 105–114, 2016.

[100] J. Broeren, K. S. Sunnerhagen, and M. Rydmark, “Haptic virtual rehabilitation in stroke: transferring research into clinical practice,” *Physical Therapy Reviews*, vol. 14, no. 5, pp. 322–335, 2009.

[101] M. A. Murphy, H. C. Persson, A. Danielsson, J. Broeren, Å. Lundgren-Nilsson, and K. S. Sunnerhagen, “Salgot-s troke a rm l ongitudinal study at the university of got henburg, prospective cohort study protocol,” *BMC neurology*, vol. 11, no. 1, p. 56, 2011.

[102] D. Swapp, V. Pawar, and C. Loscos, “Interaction with co-located haptic feedback in virtual reality,” *Virtual Reality*, vol. 10, pp. 24–30, May 2006.

[103] Y. S. Tanagho, G. L. Andriole, A. G. Paradis, K. M. Madison, G. S. Sandhu, J. E. Varela, and B. M. Benway, “2d versus 3d visualization: impact on laparoscopic proficiency using the fundamentals of laparoscopic surgery skill set,” *Journal of Laparoendoscopic & Advanced Surgical Techniques*, vol. 22, no. 9, pp. 865–870, 2012.

[104] P. Storz, G. F. Buess, W. Kunert, and A. Kirschniak, “3d hd versus 2d hd: surgical task efficiency in standardised phantom tasks,” *Surgical endoscopy*, vol. 26, no. 5, pp. 1454–1460, 2012.

[105] H. Benko, R. Jota, and A. Wilson, "Miragetable: freehand interaction on a projected augmented reality tabletop," in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 199–208, ACM, 2012.

[106] B. Jones, R. Sodhi, M. Murdock, R. Mehra, H. Benko, A. Wilson, E. Ofek, B. MacIntyre, N. Raghuvanshi, and L. Shapira, "Roomalive: Magical experiences enabled by scalable, adaptive projector-camera units," in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, (New York, NY, USA), pp. 637–644, ACM, 2014.

[107] R. Raskar, J. Van Baar, T. Willwacher, and S. Rao, "Quadric transfer for immersive curved screen displays," in *Computer Graphics Forum*, vol. 23, pp. 451–460, Wiley Online Library, 2004.

[108] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[109] E. R. Girden, *ANOVA: Repeated measures.* No. 84, Sage, 1992.

[110] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the royal statistical society. Series B (Methodological)*, pp. 289–300, 1995.

[111] D. P. Gross and M. F. Reneman, *Functional Capacity Evaluation*, pp. 1–4. New York, NY: Springer New York, 2017.

[112] "Functional Capacity Evaluation." https://www.aota.org/About-Occupational-Therapy/Professionals/WI/Capacity-Eval.aspx.

[113] "BTE EvalTech." https://www.btetech.com/product/evaltech/.

[114] "Ergos II Work Simulator."
http://www.simwork.com/Products/ErgosIIWorkSimulator.aspx.

[115] D. Peppers *et al.*, "Influence of functional capacity evaluation on physician's assessment of physical capacity of veterans with chronic pain: a retrospective analysis," *PM&R*, vol. 9, no. 7, pp. 652–659, 2017.

[116] D. P. Gross *et al.*, "A cluster randomized clinical trial comparing functional capacity evaluation and functional interviewing as components of occupational rehabilitation programs," *Journal of occupational rehabilitation*, vol. 24, no. 4, pp. 617–630, 2014.

[117] D. P. Gross, M. C. Battié, and J. D. Cassidy, "The prognostic value of functional capacity evaluation in patients with chronic low back pain: part 1: timely return to work," *Spine*, vol. 29, no. 8, pp. 914–919, 2004.

[118] F. Schaafsma *et al.*, "Physical conditioning programs for improving work outcomes in workers with back pain," *Cochrane Database Syst Rev*, vol. 1, 2010.

[119] C. James, M. Reneman, and D. Gross, "Functional capacity evaluation research: Report from the second international functional capacity evaluation research meeting," *Journal of occupational rehabilitation*, vol. 26, no. 1, pp. 80–83, 2016.

[120] F. Yakub, A. Z. M. Khudzari, and Y. Mori, "Recent trends for practical rehabilitation robotics, current challenges and the future," *International Journal of Rehabilitation Research*, vol. 37, no. 1, pp. 9–21, 2014.

[121] B. Taylor, M. E. Cupo, and S. J. Sheredos, "Workstation robotics: a pilot study of a desktop vocational assistant robot," *American Journal of Occupational Therapy*, vol. 47, no. 11, pp. 1009–1013, 1993.

[122] J. L. Schuyler and R. M. Mahoney, "Assessing human-robotic performance for vocational placement," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 3, pp. 394–404, 2000.

[123] H. M. Van der Loos, D. J. Reinkensmeyer, and E. Guglielmelli, "Rehabilitation and health care robotics," in *Springer handbook of robotics*, pp. 1685–1728, Springer, 2016.

[124] "BTE Eccentron." https://www.btetech.com/product/eccentron/.

[125] R. Ocampo and M. Tavakoli, "Improving user performance in haptics-based rehabilitation exercises by colocation of user's visual and motor axes via a three-dimensional augmented-reality display," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 438–444, 2019.

[126] J. Fong and M. Tavakoli, "Kinesthetic teaching of a therapist's behavior to a rehabilitation robot," in *2018 International Symposium on Medical Robotics (ISMR)*, pp. 1–6, March 2018.

[127] C. Martínez and M. Tavakoli, "Learning and robotic imitation of therapist's motion and force for post-disability rehabilitation," in *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*, pp. 2225–2230, IEEE, 2017.

[128] N. Hogan and S. P. Buerger, "Impedance and interaction control," in *Robotics and automation handbook* (T. R. Kurfess, ed.), ch. 19, CRC press, 2004.

[129] F. Dimeas and N. Aspragathos, "Online stability in human-robot cooperation with admittance control," *IEEE Transactions on Haptics*, vol. 9, pp. 267–278, April 2016.

[130] R. Tao, "Haptic teleoperation based rehabilitation systems for task-oriented therapy," Master's thesis, University of Alberta, 2015.

[131] J. J. Abbott, P. Marayong, and A. M. Okamura, "Haptic virtual fixtures for robot-assisted manipulation," in *Robotics research*, pp. 49–64, Springer, 2007.

[132] G. Fasano and A. Franceschini, “A multidimensional version of the kolmogorov–smirnov test,” *Monthly Notices of the Royal Astronomical Society*, vol. 225, no. 1, pp. 155–170, 1987.

[133] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, “Numerical recipes in c,” *Cambridge University Press*, vol. 1, p. 3, 1988.

[134] “Ink Painter.” https://assetstore.unity.com/packages/tools/particles-effects/ink-painter-86210.

Appendix A

2D AR System Development in Unity

A.1 Overview of the System

The setup uses two main computers, each one handles a certain area of the system. The rehabilitation robot is connected to the Quanser computer and is controlled by Matlab/Simulink software. The game environment is developed in the Unity computer using the Unity Game Engine and uses the projector to display the game to the user. The process for moving the circular avatar in Unity is as follows: the user moves the end-effector of the rehabilitation robot in which its position data is captured by the Matlab/Simulink software and sent to Unity. A calibration script in Unity converts the robot position coordinates to screen coordinates to display it properly using the projector. When the circular avatar collides with the car in the game environment, the collision information (normal vector of the contact point) is sent by Unity to the Matlab/Simulink computer to move the rehabilitation robot for force feedback.

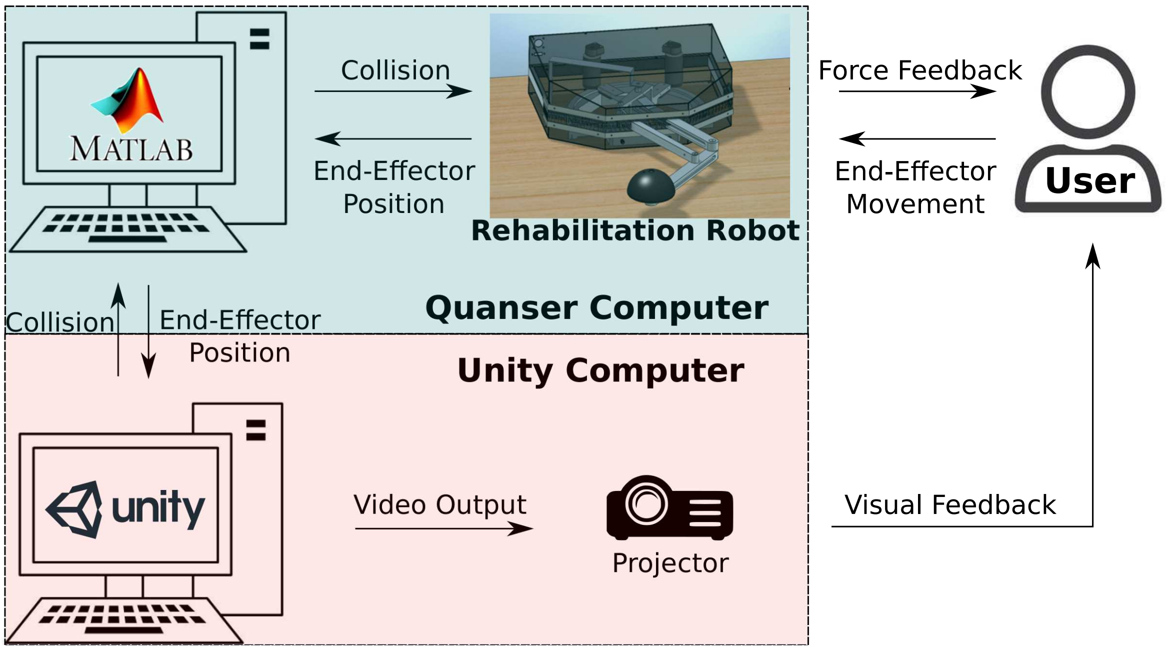
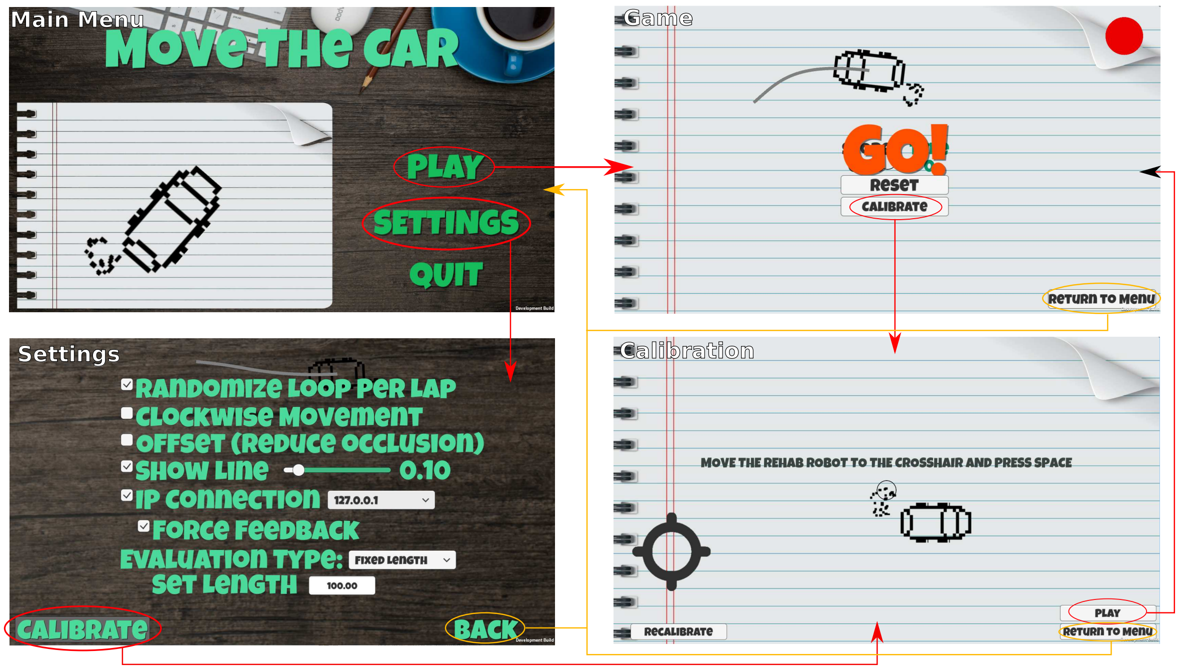


Figure A.1: The overall process of how each individual system interacts with each other for the 2D AR system.



☑ Randomize Loop Per Lap
☐ Clockwise Movement
☐ Offset (Reduce Occlusion)
☑ Show Line 0.10
☑ IP Connection 127.0.0.1
☑ Force Feedback
Evaluation Type: Fixed Length
Set Length 100.00

Figure A.2: The flowchart of how to navigate the graphical user interface for the 2D AR game. Top Left: Main menu. Top Right: Game. Bottom Left: Settings. Bottom Right: Calibration

A.2 Layout of the Game User Interface

Main Menu

Three options are given in the main menu: Play, Settings, and Quit. ***Play*** brings the user to the main task of the game. ***Settings*** leads to the settings screen that allows for customization of the game. ***Quit*** exits the game.

Settings

There are eight customizable options found in settings. ***Randomize Loop per Lap*** produces a random circular trajectory for the car to travel on each time the car finishes a lap. ***Clockwise Movement*** toggles either clockwise movement of the car along the trajectory when checked or counter-clockwise when unchecked. ***Offset*** adds a set offset to the circular object that collides with the car. This is intended to prevent the shadow during the 2D AR game from blocking the circle to let the users see direct contact between the circle and the car. ***Show Line*** shows a percentage (given by the slider beside the option) of the total length of the spline the car travels on. This is shown in front of the car to inform the user where to go. ***IP connection*** lets the user connect to the IP where the rehabilitation robot is located or localhost (127.0.0.1) for testing purposes. ***Force Feedback*** toggles the haptic feedback received from colliding with the car. ***Evaluation type*** can be changed to two options: *Fixed Length* or *Fixed Time*. This sets the limit of how far can travel, or how long the game runs before timing out. The specific value can be changed in following box below it labeled ***Set Length*** (changes to ***Set Time*** when evaluation type is set to fixed time). ***Calibrate*** brings the user to the calibration screen and ***Back*** returns them to the main menu.

Calibration

The calibration screen is where the calibration is done to align the movements of the rehabilitation robot with the movement of the circle the user controls on the screen. The user leads the end-effector to the crosshair shown on the screen and presses spacebar to record the point. This is done thrice more to get a total of four points required for calibration. This is also where the users can practice moving the car around by colliding with it.

Game

The game screen contains the actual game itself where the user pushes the car along the invisible track within the specified time limit or distance. ***Reset*** restarts the screen and returns the car back to its starting position. ***Calibrate*** returns the user to the calibration screen if any adjustments are needed.

A.3 C# code for the 2D AR System

While Unity is the game development engine used to create the game environment, the scripts attached to the objects were written in C#. This section presents the main C# codes used for the 2D AR System and a brief explanation of what each script does. For certain scripts, only excerpts of the code are displayed to minimize the length and improve readability. Areas of code that are removed are denoted by an ellipsis.

A.3.1 C# code for the Calibration Scene

The calibration scene allows the projector and the movement of the rehabilitation robot to match in both axes and scale. This code is based on the equations found in Chapter 3.

The steps to find the homography transformation are:

1. The user presses space on the displayed crosshair and the Update() function stores the robot position and screen position of that point (done for 4 points in total).

2. Calibrate() function is run. It uses the CreateCorrespondencePointsForH() function to build a 2 x 9 matrix for each set of robot-screen point based on Equation 3.11

and 3.12. The matrices are then combined to form an 8 x 9 matrix in which SVD is performed onto to find H.

Script: Calibration

```
public class Calibration : MonoBehaviour
{
    public static Calibration instance; // Allows easy access from other scripts.
    public GameObject calibrationCrosshair;
    public GameObject doneText;
    public GameObject instructionsText;
    public GameObject characterObject;

    private int spacePressed = 0;
    private GrabScript characterScript;
    private Vector3 aR, bR, cR, dR; // Robot positions
    private Vector3 aS, bS, cS, dS; // Screen positions
    private Vector2 firstPos = new Vector2(-7, -2); // Positions the crosshair will move to
    private Vector2 secondPos = new Vector2(-7, 2);
    private Vector2 thirdPos = new Vector2(7, 2);
    private Vector2 fourthPos = new Vector2(7, -2);
    private Vector2 offset = new Vector3(0f, 2.2f); // Offset to lower the crosshairs when handle
         offset is turned on

    private void Start()
    {
      // Initializes the positions of the crosshairs
        firstPos = (GameControl.instance.handleToggle) ? firstPos + offset : firstPos;
        secondPos = (GameControl.instance.handleToggle) ? secondPos + offset : secondPos;
        thirdPos = (GameControl.instance.handleToggle) ? thirdPos + offset : thirdPos;
        fourthPos = (GameControl.instance.handleToggle) ? fourthPos + offset : fourthPos;
        calibrationCrosshair.transform.position = firstPos;
        characterScript = characterObject.GetComponent<GrabScript>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown("space"))
        {
            spacePressed++;
            switch (spacePressed)
            { // Captures the 4 points needed for calibration
                case 1:
                    aR = new Vector3(MatlabServer.instance.xMove, MatlabServer.instance.yMove, 0); //
                         Record 1st robot pos
                    aS = new Vector3(calibrationCrosshair.transform.position.x,
                         calibrationCrosshair.transform.position.y, 0); // Record 1st screen pos
                    calibrationCrosshair.transform.position = secondPos; // Move crosshair to 2nd pos
                    instructionsText.SetActive(false); // Turn off instructions
                    break;
                case 2:
                    bR = new Vector3(MatlabServer.instance.xMove, MatlabServer.instance.yMove, 0); //
                         Record 2nd robot pos
                    bS = new Vector3(calibrationCrosshair.transform.position.x,
                         calibrationCrosshair.transform.position.y, 0); // Record 2nd screen pos
                    calibrationCrosshair.transform.position = thirdPos; // Move crosshair to 3rd pos
                    break;
                case 3:
```

```
                cR = new Vector3(MatlabServer.instance.xMove, MatlabServer.instance.yMove, 0); //
                     Record 3rd robot pos
                cS = new Vector3(calibrationCrosshair.transform.position.x,
                     calibrationCrosshair.transform.position.y, 0); // Record 3rd screen pos
                calibrationCrosshair.transform.position = fourthPos; // Move crosshair to 4th pos
                break;
            case 4:
                dR = new Vector3(MatlabServer.instance.xMove, MatlabServer.instance.yMove, 0); //
                     Record 4th robot pos
                dS = new Vector3(calibrationCrosshair.transform.position.x,
                     calibrationCrosshair.transform.position.y, 0); // Record 4th screen pos
                calibrationCrosshair.SetActive(false);
                Calibrate();
                characterScript.Calibrate(); // Run the calibration function
                doneText.SetActive(true);
                break;
            default:
                // Do nothing
                break;
        }
    }
}

public void Recalibrate() // Reinitialize the calibration to redo the process
{
    calibrationCrosshair.SetActive(true);
    doneText.SetActive(false);
    instructionsText.SetActive(true);
    spacePressed = 0;
    calibrationCrosshair.transform.position = new Vector3(-4f, -2f, 0);
}

private void Calibrate()
{
    // Homography DLT method
    // Each matrix is 2 x 9
    Matrix<float> point1 = Matrix<float>.Build.DenseOfArray(CreateCorrespondencePointsForH(aR, aS));
    Matrix<float> point2 = Matrix<float>.Build.DenseOfArray(CreateCorrespondencePointsForH(bR, bS));
    Matrix<float> point3 = Matrix<float>.Build.DenseOfArray(CreateCorrespondencePointsForH(cR, cS));
    Matrix<float> point4 = Matrix<float>.Build.DenseOfArray(CreateCorrespondencePointsForH(dR, dS));

    // Build 8 x 9 matrix
    Matrix<float> combinedPoints = point1.Stack(point2).Stack(point3).Stack(point4);
    Svd<float> svd = combinedPoints.Svd(); // Perform SVD on the matrix
    Matrix<float> Vmatrix = -svd.VT.Transpose(); // Transpose V matrix
  // H is the last column of V
    float[,] Hmatrix = { { Vmatrix[0, 8], Vmatrix[1, 8], Vmatrix[2, 8] },
                        { Vmatrix[3, 8], Vmatrix[4, 8], Vmatrix[5, 8]},
                        { Vmatrix[6, 8], Vmatrix[7, 8], Vmatrix[8, 8]} };
  // Save the Hmatrix so the game scene can reference it
    PlayerPrefs.SetFloat("T11", Vmatrix[0, 8]);
    PlayerPrefs.SetFloat("T12", Vmatrix[1, 8]);
    PlayerPrefs.SetFloat("T13", Vmatrix[2, 8]);

    PlayerPrefs.SetFloat("T21", Vmatrix[3, 8]);
    PlayerPrefs.SetFloat("T22", Vmatrix[4, 8]);
    PlayerPrefs.SetFloat("T23", Vmatrix[5, 8]);

    PlayerPrefs.SetFloat("T31", Vmatrix[6, 8]);
    PlayerPrefs.SetFloat("T32", Vmatrix[7, 8]);
    PlayerPrefs.SetFloat("T33", Vmatrix[8, 8]);
}

private float[,] CreateCorrespondencePointsForH(Vector3 robotPoint, Vector3 screenPoint)
{
    // Function to create a 2 x 9 matrix from each set of robot-screen point
       float[,] pointHomography = { {-robotPoint.x, -robotPoint.y, -1, 0, 0, 0,
            robotPoint.x*screenPoint.x, robotPoint.y*screenPoint.x, screenPoint.x },
                                    {0, 0, 0, -robotPoint.x, -robotPoint.y, -1,
                                         robotPoint.x*screenPoint.y, robotPoint.y*screenPoint.y,
                                         screenPoint.y} };
       return pointHomography;
    }
}
```

A.3.2 C# code for the Game Scene

The game scene is the environment where the user does the task of pushing the car along a track. It is comprised of five main scripts:

- **GameControl**: Oversees all aspects of the game
- **MatlabServer**: Connects the game to Simulink in the Quanser Computer
- **BezierSpline**: Creates the Bezier spline track
- **SplineForce**: Handles all car movement and interactions
- **GrabScript**: Handles all circular avatar movement and interactions

These scripts are described more in detail below.

Script: Game Control

The Game Control script handles the overall flow of the game. It grabs the settings chosen by the user from the settings screen and is referenced by all the other scripts. The timer, red blinker, and data collection are taken care of by the Game Control script along with starting and ending the game.

```
public class GameControl : MonoBehaviour
{
    //Main script that oversees all aspects of the game
    public static GameControl instance; // Allows easy access from other scripts. They just have to do
         GameControl.instance.something

   ...

    // Use this for initialization
    void Awake() // Initialize Settings variables
    { //Always called before start() functions
      //Makes sure that there is only one instance of GameControl (singleton)
        if (instance == null) //If no game control found
    {
        instance = this; //Then this is the instance of the game control
        sceneIndex = (SceneName)SceneManager.GetActiveScene().buildIndex;
        isRehab = PlayerPrefs.GetInt("RehabToggle", 0) == 1 ? true : false;
        randomize = PlayerPrefs.GetInt("RandomizeToggle", 1) == 1 ? true : false;
        forceFeedback = PlayerPrefs.GetInt("ForceToggle", 1) == 1 ? true : false;
        isCW = PlayerPrefs.GetInt("DirectionToggle", 1) == 1 ? true : false;
        handleToggle = PlayerPrefs.GetInt("HandleToggle", 1) == 1 ? true : false;
        evalType = PlayerPrefs.GetInt("EvalType", 0);
        DetermineCharacter();
        goal = (evalType == 0) ? PlayerPrefs.GetFloat("Length", 40f) : PlayerPrefs.GetFloat("Time",
             30f);
        if (sceneIndex == SceneName.Main) calibrateButton.SetActive(isRehab); //If main game, turn
             on blinker
        blinkerTime = blinkTimerOn;
    }
    else if (instance != this) //If the game object finds that instance is already on another game
         object, then this destroys itself as it's not needed
    {
        Destroy(gameObject);
    }
}

private void Start()
{
    if (isRehab && MatlabServer.instance.serverRunning == false)
    {
        MatlabServer.instance.StartThread();
    }
    carScript = car.GetComponent<SplineForce>();
}

// Update is called once per frame
void Update()
{
    if (sceneIndex == SceneName.Main && !gameOver)
    {
        if (!gameStart) //During countdown
        {
            timeLeft -= Time.deltaTime;
            viewedTime = timeLeft - 1;
            if (viewedTime < -1) //Start of game
            {
                gameStart = true;
                countdownText.enabled = false;
                SaveToExcel.instance.parameters.Add(forceFeedback ? 1:0);
            }
            else if (viewedTime > 0.5)
            {
                countdownText.text = viewedTime.ToString("F0");

            }
            else
            {
                countdownText.text = "GO!";
            }
        }
        else //When game starts
        {
            blinkerTime -= Time.deltaTime; //For blinker
            if (blinkerTime < 0 && blinker.enabled)
            {
                blinker.enabled = false;
                blinkerTime = blinkTimerOff;
            }
            else if (blinkerTime < 0 && !blinker.enabled)
            {
                blinker.enabled = true;
                blinkerTime = blinkTimerOn;
            }
            timeElapsed += Time.deltaTime; //For time score
            SaveToExcel.instance.timeElapsed.Add(timeElapsed);
            SaveToExcel.instance.characterPositionX.Add(characterController.transform.position.x);
            SaveToExcel.instance.characterPositionY.Add(characterController.transform.position.y);
            SaveToExcel.instance.carPositionX.Add(car.transform.position.x);
            SaveToExcel.instance.carPositionY.Add(car.transform.position.y);
            SaveToExcel.instance.lap.Add(car.GetComponent<SplineForce>().GetLap);
            timeText.text = "Time \n" + timeElapsed.ToString("F1");
        }
        CheckGameEnd();
    }
}

public void Restart()
{
    if (isRehab)
    {
        MatlabServer.instance.StopThread();
    }
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex); //Reload current scene to
         restart
}
public void Calibrate()
{
    if (GameControl.instance.isRehab)
    {
        MatlabServer.instance.StopThread();
    }
    SceneManager.LoadScene("Calibrate"); //Load scene
}

private void CheckGameEnd()
{
    if (evalType == 0) //Length
    {
        if (carScript.GetScore >= goal)
        {
            GameOver();
        }
    }
    else if (evalType == 1) //Time
    {
        if (timeElapsed >= goal)
        {
            GameOver();
        }
    }
    else
    {
     //Do nothing
    }
}

private void GameOver() // Function to put the game in GameOver state
{
    gameOver = true;
    gameStart = false;
    carScript.SetCurrentPosition(carScript.GetCurrentPosition); //Stop it at current position
    car.GetComponent<Rigidbody2D>().velocity = Vector2.zero; //Remove any velocity
    gameOverText.enabled = true;
    }

    private void DetermineCharacter() // Used if the "Handle Offset" in Settings is turned on
    {
        if (handleToggle)
        {
            handleOffset.SetActive(true);
            characterController.SetActive(false);
        }
        else
        {
            handleOffset.SetActive(false);
            characterController.SetActive(true);
        }
    }
}
```

Script: Matlab Server

Matlab Server is the main communication script that allows the Rehab robot and Unity to share information with each other. The Rehab robot sends Unity the position data of the end-effector, and Unity sends back collision information.

```
public class MatlabServer : MonoBehaviour {
   //This code connects the unity program with matlab to acquire end-effector data
    public static MatlabServer instance;
    public float xMove, yMove = 0;
    public float xForce, yForce = 0;
    public float collisionStatus = 0;
    public float forceFeedback = 0;
    [ReadOnly] public bool serverRunning = false;
    [ReadOnly] public string ipAddress = "127.0.0.1"; //This comp: ***.***.**.**, Localhost: 127.0.0.1
    [ReadOnly] public int port = 9000;

    private Thread thread;
    private Socket newsock;
    private bool stop = false;

    // Use this for initialization
    void Awake()
    { //Always called before start() functions
      //Makes sure that there is only one instance of Matlab Server (singleton)
        if (instance == null) //If no game control found
        {
            instance = this; //Then this is the instance of the game control
        }
        else if (instance != this) //If the game object finds that instance is already on another game
             object, then this destroys itself as it's not needed
        {
            Destroy(gameObject);
        }
        ipAddress = PlayerPrefs.GetString("IPAddress", "127.0.0.1");
        forceFeedback = (float)PlayerPrefs.GetInt("ForceToggle", 1);
    }
```

```
void OnApplicationQuit()
{
    if (GameControl.instance.isRehab)
    {
        Debug.Log("Quit~!");
        thread.Abort();
        newsock.Close();
    }
}

public void StartThread()
{
    thread = new Thread(new ThreadStart(ThreadMethod));
    thread.Start();
}

public void StopThread()
{
    if (serverRunning)
    {
        stop = true;
    }
    else
    {
        thread.Abort();
    }
    newsock.Close();
}

private void ThreadMethod()
{
    int recv;
    byte[] dataRecv = new byte[16]; //Data Received from Simulink
    byte[] dataSend = new byte[32]; //Send Collision Status, X, Y
    IEnumerable<byte> dataSendLINQ = new byte[32]; //Initialize LINQ for easy concatenation later
         for sending
    //Create IP End point, where I want to connect (Local IP/Port)
    IPEndPoint ipep = new IPEndPoint(IPAddress.Parse(ipAddress), port);
    //Create UDP Socket
    newsock = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
    //Bind to ip. Server waits for a client at specified ip & port.
    try
    {
        newsock.Bind(ipep);
    }
    catch (Exception e)
    {
        Debug.Log("Winsock Error: " + e.ToString());
    }
    Debug.Log("Connecting to IP: "+ ipAddress + " Port: "+ port +" Waiting for a client...");

    //Get IP of client
    IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
    EndPoint Remote = (EndPoint)(sender);

    //Receive binary Data from client
    recv = newsock.ReceiveFrom(dataRecv, ref Remote);

    //Decode data and display
    Debug.Log("Message received from " + Remote.ToString());
    serverRunning = true;

    while (true)
    {
        //Receive X Y positions
```

```
            dataRecv = new byte[16];
            recv = newsock.ReceiveFrom(dataRecv, ref Remote);

            //Convert Bytes into Doubles (This gets referenced by mainPlayer.cs to move the character)
                 X/Y Position
            xMove = (float)BitConverter.ToDouble(dataRecv, 0); //x
            yMove = (float)BitConverter.ToDouble(dataRecv, 8); //y

            //Concatenate Collision Status, ForceFeedbackStatus, xForce, yForce.
            dataSendLINQ = (BitConverter.GetBytes((double)collisionStatus))
               .Concat(BitConverter.GetBytes((double)forceFeedback))
               .Concat(BitConverter.GetBytes((double)xForce))
               .Concat(BitConverter.GetBytes((double)yForce));
            dataSend = dataSendLINQ.ToArray(); //Convert to byte Array from IEnumerable byte Array

            //Send Info to Simulink
            newsock.SendTo(dataSend, dataSend.Length, SocketFlags.None, Remote);

            if (stop)
            {
                break;
            }
        }
        Debug.Log("Exiting Thread...");
    }
}
```

Script: Bezier Spline

This script handles the creation of the Bezier spline that is used for the track that the car is restrained to. The actual code for creating the Bezier spline is excluded from the script to minimize the length, however, it can be found in the link in the commented part. Below are the functions written to snap the car onto the spline. The function divides the whole path into multiple steps and iterates through each step to find the closest point. It then cuts a segment around the point in which it then subdivides the interval into multiple steps once more to find the closest point. This happens five times in total.

```
public class BezierSpline : MonoBehaviour {
   // This script is the basis for creating a bezier spline for the car's track.
   // Code here is based upon work by: https://catlikecoding.com/unity/tutorials/curves-and-splines/
    ...
   // The following functions are added to snap the car onto the closest point on the spline
    public float ClosestBezierTime(Vector3 point) // Returns t of bezier where point in worldspace is
         closest to (multiple iterations)
    {
        float t = BestFitTime(point, 0, 1, 100);
        float delta = 1.0f / 10.0f;
        for (int i = 0; i < 4; i++)
    {
        t = BestFitTime(point, t - delta, t + delta, 100);
        delta /= 10;
    }
    return t;
}

public float BestFitTime(Vector3 point, float start, float end, int steps) //Calculate best fit
     time in given interval
{
    start = Mathf.Clamp01(start);
    end = Mathf.Clamp01(end);
    float step = (end - start) / (float)steps;
    float Res = 0;
    float Ref = float.MaxValue; // Will be replaced by distance of bezier(t) to point
    for (int i = 0; i < steps; i++)
    {
        float t = start + step * i;
        float L = (GetPoint(t) - point).sqrMagnitude;
        if (L < Ref)
        {
            Ref = L;
            Res = t;
        }
    }
    return Res;
}

public Vector3 ClosestBezierPoint(Vector3 point)
{
    return GetPoint(ClosestBezierTime(point));
}

public Vector3 OnLine(Vector3 point) // If on line, return 0, else return point closest to line
{
    Vector3 closestPoint = ClosestBezierPoint(point);
    if (closestPoint == point)
    {
        return Vector3.zero;
    }
    return closestPoint;
}

public Vector3 GetPoint(float t) { // Gets point in total spline corresponding to [0,1]
    int i;
    if (t >= 1f) {
        t = 1f;
        i = points.Length - 4;
    }
    else {
        t = Mathf.Clamp01(t) * CurveCount;
        i = (int)t;
        t -= i;
        i *= 3;
    }
    return transform.TransformPoint(Bezier.GetPoint(points[i], points[i + 1], points[i + 2],
         points[i + 3], t));
}

public float SplineLength() // Returns spline total length
{
    float length = 0;
    float start = 0;
    float steps = 100f; // Amount of times we'll divide the spline
    for (float i = 1f; i <= steps; i++)
```

```
        {
            length += (GetPoint(i / steps) - GetPoint(start)).magnitude;
            start = i / steps;
        }
        return length;
    }
    public float CurrentLength(float t) //Returns current length travelled through spline from start
    {
        float length = 0 , start = 0 , next = 0;
        float end = t;
        float steps = 100f; // Amount of times we'll divide the spline
        float step = end / steps;

        for (float i = 1f; i <= steps; i++)
        {
            next = step * i;
            length += (GetPoint(next) - GetPoint(start)).magnitude;
            start = next;
        }
        return length;
    }
```

Script: Spline Force

This script handles everything that involves the movement of the car such as collisions with the end-effector, progress of the car along the track, and proper car rotation. It also uses the Bezier Spline script discussed previously to restrain the car on the track.

```
public class SplineForce : MonoBehaviour
{
    //This script handles the collisions experienced by the car and ensures it stays on the track
    //Also keeps track of how far the car is in the track

   ...

    private void Start()
    {
        spline = (GameControl.instance.isCW) ? splineCW : splineCCW; // CW or CCW movement
        rb2d = GetComponent<Rigidbody2D>(); // Get rigidbody of car
        rb2d.velocity = Vector2.zero; // Set velocity to 0
        transform.localPosition = spline.GetPoint(0); // Move car to start of spline
        currentPosition = transform.localPosition; // Save current position
        randomizerScript = spline.GetComponent<SplineRandomizer>(); // Randomize spline shape
    }

    private void Update() // Uses the functions listed below to update the car frame by frame
    {
        // Move method
        if (!GameControl.instance.gameStart || GameControl.instance.gameOver || goingBackwards)
        { // Prevents car from moving during countdown or gameover
            transform.localPosition = currentPosition; // Locks car in current position
        }
        else
        {
            MoveBySnappingToCurve(); // Runs function to snap to curve
        }
        UpdateScore(); // Runs function to update score
        CarRotation(); // Runs function to rotate car to the right orientation
    }

    private void OnCollisionEnter2D(Collision2D collision) // During collision with end-effector
    {
        Vector3 contactNormal = collision.contacts[0].normal.normalized; // Gives unit vector direction
             normal to point of contact
        float dotProduct = Vector3.Dot(contactNormal, spline.GetDirection(progress));
        if (!goingBackwards) // Prevents movement of car backwards
        {
            currentPosition = transform.localPosition;
        }
        goingBackwards = (dotProduct < 0) ? true : false;
    }

    private void CarRotation() // Used to make sure sprite is looking at the right direction
    {
        Vector3 diff = spline.GetDirection(progress); // Get direction based on progress
        float rot_z = Mathf.Atan2(diff.y, diff.x) * Mathf.Rad2Deg;
        transform.rotation = Quaternion.Euler(0f, 0f, rot_z);
    }

    private void MoveBySnappingToCurve() // Restricts the car to spline
    {
        progress = spline.ClosestBezierTime(transform.position); // Finds at what point t [0,1] the car
             is at
        Vector3 position = spline.GetPoint(progress); // Finds the location on the spline t is

        if ((prevProgress - progress) > 0.9 && mode == SplineWalkerMode.Loop) //If there's a big jump
             in progress (going from current loop to the next)
        {
            lap++;
            lengthSum += spline.SplineLength(); //total length
            if (GameControl.instance.randomize) // Randomize spline after each lap
            {
                randomizerScript.Randomize(true);
            }
        }
        if (position != transform.position) //If the position is not equal to current position, move to
             position
        {
            transform.localPosition = position;
            prevProgress = progress;
        }
    }

    private void UpdateScore() // Updates total distance traveled
    {
        score = lengthSum + spline.CurrentLength(progress);
        scoreText.text = " Score \n" + score.ToString("F2");
    }
}
```

Script: Grab Script

This script handles the movement of the circular avatar. The calibration parameters saved by the Calibrate script are referenced and used to convert the robot coordinates to screen coordinates. Force feedback due to collision with the car is also sent to the end-effector for the user to feel.

```
public class GrabScript : MonoBehaviour {
    //Moves the circular avatar to the position of the end-effector

    ...

    private void Start()
    {
        Calibrate(); // Get saved calibration data
        carRb2d = car.GetComponent<Rigidbody2D>(); // Get rigidbody of car
        forceFeedback = GameControl.instance.forceFeedback; // If haptics ON in settings
        characterTransform = (GameControl.instance.handleToggle) ? transform.parent.transform :
             transform; //Get parent transform if handle offset is on. If not, get object transform
    }

    public void Calibrate() //Initialize position from saved calibration matrix
    {
        try
        {
            T11 = PlayerPrefs.GetFloat("T11");
            T12 = PlayerPrefs.GetFloat("T12");
            T13 = PlayerPrefs.GetFloat("T13");

            T21 = PlayerPrefs.GetFloat("T21");
            T22 = PlayerPrefs.GetFloat("T22");
            T23 = PlayerPrefs.GetFloat("T23");

            T31 = PlayerPrefs.GetFloat("T31");
            T32 = PlayerPrefs.GetFloat("T32");
            T33 = PlayerPrefs.GetFloat("T33");
        }
        catch
        {
            Debug.Log("Please Calibrate");
        }
    }

    // Update is called once per frame
    void FixedUpdate ()
    {
        ...
        characterTransform.position = CalibratedMovement();
        ...
    }

    private void OnCollisionEnter2D(Collision2D collision)
    {
      //Gives unit vector direction normal to point of contact
        Vector3 contactNormal = collision.contacts[0].normal.normalized;
        //Forces experienced by player from contact
        xForce = contactNormal.x * carRb2d.mass;
        yForce = contactNormal.y * carRb2d.mass;
    }

    private void OnCollisionStay2D(Collision2D collision) // While in contact with ball
    {
      //Gives unit vector direction normal to point of contact
        Vector3 contactNormal = collision.contacts[0].normal.normalized;
        if (forceFeedback) // If haptics is ON
        {
         //Forces experienced by player from contact
            xForce = contactNormal.x * carRb2d.mass;
            yForce = contactNormal.y * carRb2d.mass;
        }
        else // Haptics is OFF
        {
            xForce = 0;
            yForce = 0;
        }
        MatlabServer.instance.collisionStatus = 1f;
        MatlabServer.instance.xForce = xForce;
        MatlabServer.instance.yForce = yForce;
    }
    private void OnCollisionExit2D(Collision2D collision)
    {
      // Reset forces to 0 after exiting collision
        xForce = 0;
        yForce = 0;
        MatlabServer.instance.collisionStatus = 0f;
        MatlabServer.instance.xForce = xForce;
        MatlabServer.instance.yForce = yForce;
    }

    public Vector3 CalibratedMovement() // Converts Robot points to Screen points
    {
        // Ps = T*Pr
        float xS = MatlabServer.instance.xMove * T11 + MatlabServer.instance.yMove * T12;
        float yS = MatlabServer.instance.xMove * T21 + MatlabServer.instance.yMove * T22;
        float lambda = MatlabServer.instance.xMove * T31 + MatlabServer.instance.yMove * T32 + T33;
        return new Vector3(xS/lambda, yS/lambda, 0f);
    }
}
```

Appendix B

3D AR System Development in Unity

B.1 Layout Overview of Game

The system is comprised of two main areas: the Quanser computer, and the Unity computer. The Quanser computer controls the HD^2 Robot using MATLAB and Simulink. It receives information from the HD^2 about the end-effector's position and orientation. The Unity Computer houses the game developed using the Unity Game Engine and C#, utilizes the Kinect for head-tracking, and displays the game to the user through the projector. The system is split into two areas to decrease the load on each computer and instead, the two computers communicate through a User Datagram Protocol (UDP) connection.

The user's movement of the end-effector of the HD^2 is captured by Simulink and then sent to the Unity computer. This information is used to move the user's cursor (e.g. small sphere/hoop/ball depending on the task) in the 3D environment. The projector displays the game in front of the user. The position of the user's head is taken by the Kinect to adjust the user's point of view in the 3D environment to display the correct perspective. Compared to the 2D system, the user is required to use the

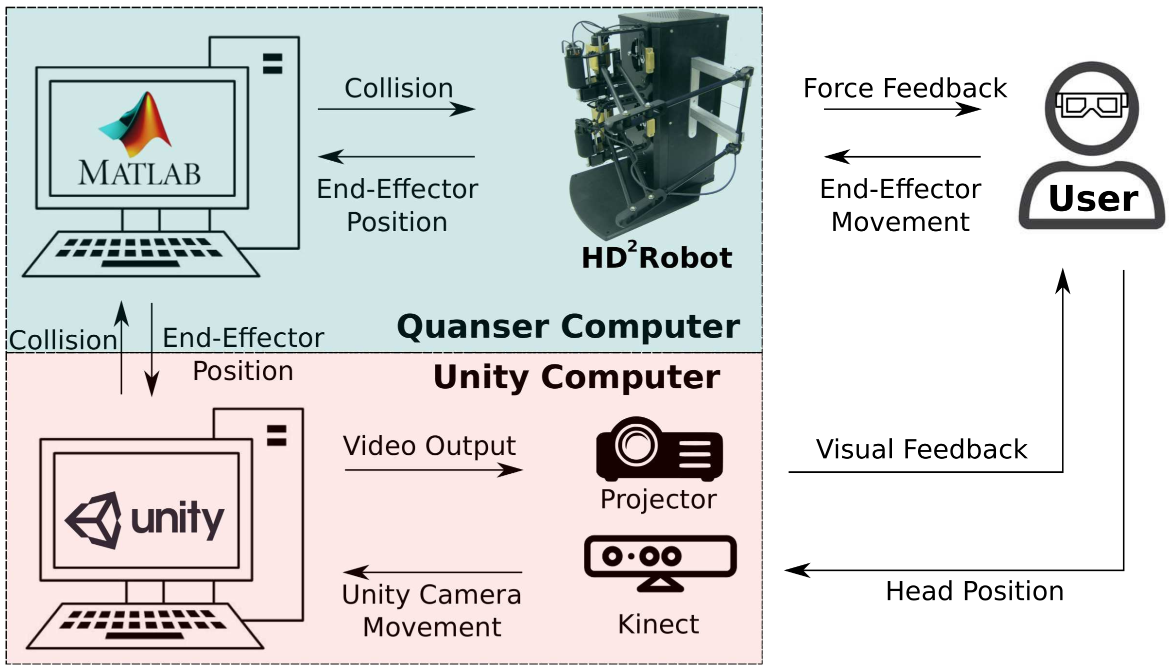


Figure B.1: The overall process of how each individual system interacts with each other for the 3D AR system.

DLP Link 3D shutter glasses to properly see 3D. Collisions in the game are detected by Unity and sent to the Quanser computer to give the user haptic feedback. Since the virtual objects (e.g. balls that bounce off the hoop) have no actual mass, the feedback is tuned by trial-and-error.

B.2 C# Code for the Three Tasks

The scripts used for this work are mainly derived from the codes used in Chapter 3, namely, **GameControl**, **MatlabServer**, and **GrabScript**. Since minor changes were done to integrate these scripts into this 3D system, they will not be described. This system does not have a graphical user interface. The settings were configured by changing the values in a text file for the application to read at start-up. In order to improve readability and decrease the amount of code, only the code for the snapping task will be shown. These codes aim to show the general idea behind the development of the other tasks in which there are spawner scripts to instantiate the holes and the spheres and the object scripts that attach directly to the instantiated holes and spheres that handle the interaction with the end-effector.

B.2.1 Code for Snapping

The main codes in the snapping task involve the creation of the spheres that the end-effector snaps on to, and the interaction of these spheres with the end-effector to allow for snapping.

Script: GravityWellSpawner

This script spawns a specified number of spheres randomly around the workspace that the end-effector can snap on to. It can also pre-generate a table of spawn points which it can load later to allow for the presentation of the same set of spawn points to different users. This script also handles highlighting of the spheres and increasing/decreasing user scores.

```
public class GravityWellSpawner : MonoBehaviour {
  ...
   // Use this for initialization
   void Start () {
       GameControl.instance.sceneNumber = 2; // Change scene number on Game Control
       spawnedBalls = new GameObject[initialSpawnCount]; // Initialize array of balls
       // Instantiate initial point to reach
       initialStart = new Vector3(-0.0523f, -0.0542f, 0.8347f); //Initial point
       spawnedBalls[0] = Instantiate(gravityWellPrefab, initialStart, Quaternion.identity); //First
            spawn
       rend = spawnedBalls[0].GetComponent<Renderer>();
       originalColor = rend.material.color;
       rend.material.color = Color.yellow; // Make it color yellow
       spawnedBalls[0].GetComponent<GravityWell>().highlightedBall = true; // Set its status to
            highlighted
       // Set boundaries for area of spawn
       boxSize = spawnBoxArea.GetComponent<Collider>().bounds.size;
       boxCenter = spawnBoxArea.transform.position;
       colliderRadius = gravityWellPrefab.GetComponent<SphereCollider>().radius;
       colliderScale = gravityWellPrefab.GetComponent<SphereCollider>().transform.lossyScale.x;

       if (GameControl.instance.useData)
       {
           LoadData(); // Use data from CSV
       }

       // Spawn # of balls cased on spawn count
       for (int i = 1; i < initialSpawnCount; i++)
       {
           SpawnGravityWell(i);
```

```
            if (GameControl.instance.useData)
            {
                randomizeList.Add(int.Parse(loadedData[3][i-1]));
            }
            else
            {
                randomizeList.Add(i);
            }
        }

        if (!GameControl.instance.useData) ListShuffle.Shuffle<int>(randomizeList); //Shuffle List
        backupRandomizedList = new List<int>(randomizeList); // Must make new instance or else backup
             will just reference randomizeList
        Debug.Log(backupRandomizedList.Count);
    }

   // Update is called once per frame
   void Update () {
        if (Input.GetKey(KeyCode.Space) && KeyInputDelayTimer + 0.1f < Time.time)
        {
            KeyInputDelayTimer = Time.time;
            HighlightRandomBall();
        }
        if (Input.GetKey(KeyCode.G) && KeyInputDelayTimer + 0.1f < Time.time)
        {// Press G to generate spawn points
            KeyInputDelayTimer = Time.time;
            GeneratePositionSets();
        }
    }

    // Spawn gravity well
    public void SpawnGravityWell(int index)
    {
        if (GameControl.instance.useData)
        {
            pos = new Vector3(float.Parse(loadedData[0][pointIndex]),
                 float.Parse(loadedData[1][pointIndex]), float.Parse(loadedData[2][pointIndex]));
            pointIndex++;
        }
        else
        {
            // Randomize location within box
            xSpawn = Random.Range(boxCenter.x - boxSize.x / 2, boxCenter.x + boxSize.x / 2);
            ySpawn = Random.Range(boxCenter.y - boxSize.y / 2, boxCenter.y + boxSize.y / 2);
            zSpawn = Random.Range(boxCenter.z - boxSize.z / 2, boxCenter.z + boxSize.z / 2);
            pos = new Vector3(xSpawn, ySpawn, zSpawn);

            // Check if spawn point will collide with other points
            while (Physics.CheckSphere(pos, colliderRadius * colliderScale, layerMask))
            {
                // Randomize again if it will collide
                xSpawn = Random.Range(boxCenter.x - boxSize.x / 2, boxCenter.x + boxSize.x / 2);
                ySpawn = Random.Range(boxCenter.y - boxSize.y / 2, boxCenter.y + boxSize.y / 2);
                zSpawn = Random.Range(boxCenter.z - boxSize.z / 2, boxCenter.z + boxSize.z / 2);
                pos = new Vector3(xSpawn, ySpawn, zSpawn);
            }
            //Save point locations
            xSpawnList.Add(xSpawn);
            ySpawnList.Add(ySpawn);
            zSpawnList.Add(zSpawn);
        }
        // Spawn point
        spawnedBalls[index] =Instantiate(gravityWellPrefab, pos, Quaternion.identity);
    }
    public Vector3 MoveGravityWell()
    {
        xSpawn = Random.Range(boxCenter.x - boxSize.x / 2, boxCenter.x + boxSize.x / 2);
        ySpawn = Random.Range(boxCenter.y - boxSize.y / 2, boxCenter.y + boxSize.y / 2);
        zSpawn = Random.Range(boxCenter.z - boxSize.z / 2, boxCenter.z + boxSize.z / 2);
        pos = new Vector3(xSpawn, ySpawn, zSpawn);
        return pos;
    }

    // Highlighting a random ball
    public void HighlightRandomBall()
    {
        spawnedBalls[randomizeList[0]].GetComponent<Renderer>().material.color = Color.yellow; // Make
            it color yellow
        spawnedBalls[randomizeList[0]].GetComponent<GravityWell>().highlightedBall = true; // Set its
            status to highlighted
        randomizeList.RemoveAt(0);
        if (randomizeList.Count == 0) // If list went empty
        {
            randomizeList = new List<int>(backupRandomizedList); // Restart from the beginning
        }
    }

    public void IncreaseScore()
    {
        hits++;
        score = hits - misses;
        scoreText.text = " Score\n" + score.ToString();
    }

    public void DecreaseScore()
    {
        misses++;
        score = hits - misses;
        scoreText.text = " Score\n" + score.ToString();
    }

    // Generate a CSV file of spawn locations. First row is X, 2nd is Y, 3rd is Z. Each column
        corresponds to a point in space
    private void GeneratePositionSets()
    {
        table.Add(xSpawnList);
        table.Add(ySpawnList);
        table.Add(zSpawnList);
        randomizeList.ForEach(i => doublesRandomizedList.Add(i));
        table.Add(doublesRandomizedList);
        SaveToExcel.instance.Save(table, 4, "Gravity_Spawn");
        table.Clear();
        xSpawnList.Clear();
        ySpawnList.Clear();
        zSpawnList.Clear();
        Debug.Log("Generated positions!");
    }

    // Load pre-generated spawn point data
    private void LoadData()
    {
        loadedData = SaveToExcel.instance.Load("SpawnGeneration/Gravity_Spawn_" +
            GameControl.instance.spawnNumber + ".csv");
        Debug.Log("Loaded Data");
    }
}
```

Script: GravityWell

This script handles the properties of each sphere spawned by the GravityWellSpawner. It sends forces to the MatlabServer when the end-effector snaps onto the sphere and saves the information of each snapping point and the movement of the end-effector in a CSV file.

```
public class GravityWell : MonoBehaviour {
  ...
   // Use this for initialization
   void Start () {
       myCollider = GetComponent<SphereCollider>(); // Get collider of this object
       normalizedGain = gain/0.01f; //
       topJoint = GameObject.Find("Top").transform; // Get the top part of the end effector
       gravitySpawnerScript =
           GameObject.Find("GravityWellSpawner").GetComponent<GravityWellSpawner>(); // Get access to
           the spawner script
       timeSpawned = Time.time; // Time this object was spawned
       ballRenderer = GetComponent<Renderer>(); // Get the renderer to control color for later
       originalColor = new Color(108/255f, 104/255f, 159/255f); // Unhighlighted color
       gravityWellPosition.Add(transform.position.x); // Save the x y z position in a table
       gravityWellPosition.Add(transform.position.y);
       gravityWellPosition.Add(transform.position.z);
   }

   // Update is called once per frame
   void Update() {
       // Checks how close the end-effector is from it
       xDiff = topJoint.position.x - transform.position.x;
       yDiff = topJoint.position.y - transform.position.y;
       zDiff = topJoint.position.z - transform.position.z;
       displacementFromTopJoint = new Vector3(xDiff, yDiff, zDiff);
       if (snap) // If the end-effector hits the object, send forces to snap
       {
           MatlabServer.instance.xFTop = -normalizedGain * displacementFromTopJoint.z;
           MatlabServer.instance.yFTop = normalizedGain * displacementFromTopJoint.x;
           MatlabServer.instance.zFTop = -normalizedGain * displacementFromTopJoint.y;
       }
   }

   // When end-effector enters the trigger
   private void OnTriggerEnter(Collider other)
   {
       if (other.gameObject.tag == "Player") { // If it was the end-effector
           snap = true; // Turn on snap
       }
       if (highlightedBall == true) // If this is the highlighted ball
       {
           highlightedBall = false; // Mark that it's not highlighted anymore
           wasHighlightedBall = true; // Mark that it was highlighted
           ballRenderer.material.color = originalColor; // Unhighlight
           if (!GameControl.instance.gameOver) gravitySpawnerScript.HighlightRandomBall(); // Stop
               highlighting if gameover
           if (!GameControl.instance.gameStart) // If game has not started, start game (for initial
               point)
           {
               GameControl.instance.gameCountdown = true;
```

```
            GameObject cylinder = GameObject.Find("CalibrationCylinder");
            cylinder.SetActive(false);
        }
        else // Increase score after game started
        {
            gravitySpawnerScript.IncreaseScore();
            timeEntered = GameControl.instance.timeElapsed; // Time user entered the point
            timeDuration = timeEntered - gravitySpawnerScript.timeLastBallLeft;
            timeDurationList.Add(timeDuration); // Time to reach this point from the previous point
        }
    }
    else
    {
        if (GameControl.instance.gameStart)
        { // Decrease score if it was an unhighlighted ball
            gravitySpawnerScript.DecreaseScore();
        }
    }
}

// When end-effector exits the trigger
private void OnTriggerExit(Collider other)
{
    if (other.gameObject.tag == "Player") // If it was the end-effector
    {
        pointSnapperScript = other.GetComponent<PointSnapper>();
        snap = false; // Turn off snap and reset forces to 0
        MatlabServer.instance.xFTop = 0;
        MatlabServer.instance.yFTop = 0;
        MatlabServer.instance.zFTop = 0;
        gravitySpawnerScript.timeLastBallLeft = GameControl.instance.timeElapsed; // Time user
             exited the point

        if (GameControl.instance.gameStart && wasHighlightedBall)
        { // If it exited a highlighted ball
            wasHighlightedBall = false;
            pointSnapperListLength = pointSnapperScript.listLength;

            timeDurationList.Add(gravitySpawnerScript.score); // Add current net score
            timeDurationList.Add(gravitySpawnerScript.hits); // Add current # of highlighted balls
                 hit
            timeDurationList.Add(gravitySpawnerScript.misses); // Add current # of unhighlighted
                 balls hit
            table.Add(timeDurationList); // Contains timeduration, score, hits, misses
            table.Add(gravityWellPosition); // contains x,y,z of current gravity well
            // Next rows add: time, x, y, z trajectories
            table.Add(pointSnapperScript.time.GetRange(pointSnapperScript.time.Count -
                 pointSnapperListLength, pointSnapperListLength));
            table.Add(pointSnapperScript.xDir.GetRange(pointSnapperScript.xDir.Count -
                 pointSnapperListLength, pointSnapperListLength));
            table.Add(pointSnapperScript.yDir.GetRange(pointSnapperScript.yDir.Count -
                 pointSnapperListLength, pointSnapperListLength));
            table.Add(pointSnapperScript.zDir.GetRange(pointSnapperScript.zDir.Count -
                 pointSnapperListLength, pointSnapperListLength));
            pointSnapperScript.listLength = 0;
            SaveToExcel.instance.Save(table, 6, "Snapping" + GameControl.instance.spawnNumber +
                 "_AV" + GameControl.instance.ARVR + "_CL" + GameControl.instance.cognitiveLoading +
                 "_Sc" + gravitySpawnerScript.hits);
            timeDurationList.Clear();
            table.Clear();
        }
    }
}
}
```

Appendix C

Motoman AR System Development in Unity

C.1 Layout Overview of Game

The system is comprised of four main areas: the Matlab computer, the Motoman computer, the Unity computer, and the Motion Tracker computer. The Matlab computer handles the admittance control for the Motoman robot. It is the main "hub" for the communication between the Motoman computer and the Unity computer. The Motoman computer runs a WinCE virtual operating system environment that communicates with the Motoman robot using C++ to make it move. The Unity computer runs the game environment that is developed in C# and projects it to the user. The Motion Tracker computer is not connected to the other systems, rather, it is only used to collect the user's arm biomechanics.

To directly manipulate the Motoman robot to move to a position, the user has to apply forces on the force sensor that is attached to the Motoman's end-effector. The force sensor sends these forces directly to the admittance controller in Simulink and calculates the velocity vector for the Motoman robot to move towards. The position of the Motoman is sent to Unity to update the position of the cursor (in this case, the

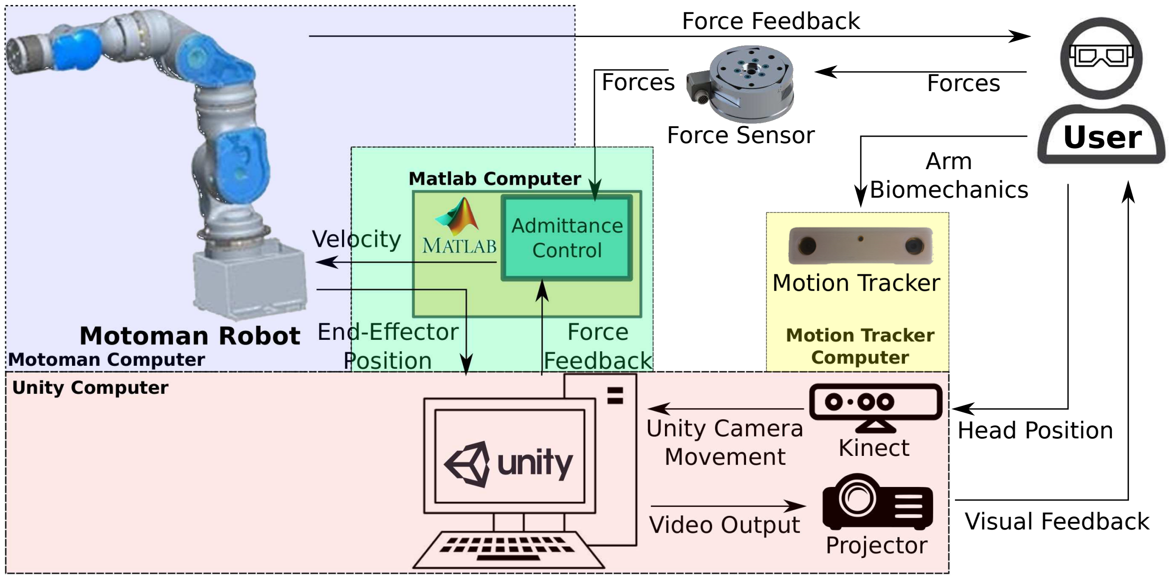


Figure C.1: The overall process of how each individual system interacts with each other for the Motoman AR system.

virtual paint roller) in the virtual environment. When the paint roller collides with the virtual wall, the direction of the collision normal (from the wall) is sent to Simulink and it is used to nullify all the forces sent to the admittance controller at that particular direction. This results in the robot not moving in towards the direction of the collision. The position of the user's head is retrieved by the Kinect to adjust the point of view shown by Unity to the correct perspective. DLP Link 3D Shutter glasses are worn by the user to see the virtual environment in 3D.

C.2 C# Code for the Painting Task

The scripts used for this work are mainly derived from the codes used in Chapter 3, namely, **GameControl**, **MatlabServer**, and **GrabScript**. Since minor changes were done to integrate these scripts into this Motoman AR system, they will not be described. This system does not have a graphical user interface. The painting task makes use of the InkPainter Asset [134] available for download in Unity.

Script: CanvasCollision.cs

This script displays the percentage of the wall filled by paint. It also signals the start and stop time for when the motion tracker camera collects data. Data collection starts once the paint roller is in contact with the wall. It stops when contact with the wall is broken after reaching ≥95%

```
public class CanvasCollision : MonoBehaviour {

   // Use this for initialization
   void Start () {
      canvasScript = GetComponent<Es.InkPainter.InkCanvas>(); // Grab attached inkpainter canvas
      fillPercent = GetComponentInChildren<TextMesh>(); // Grab the text compontent to update
      paintMaterial = GetComponent<MeshRenderer>().material.name; // Grab the material to be painted
          on
  }

   private void OnCollisionEnter(Collision collision)
   {
      if (!colorGrabbed)
      {
         Debug.Log("GameStart");
         GameControl.instance.gameStart = true; // Start game
         MatlabServer.instance.gameStart = 1; // Send to Matlab game start
         collisionPainterColor =
             collision.gameObject.GetComponent<Es.InkPainter.Sample.CollisionPainter>().Color;
         colorGrabbed = true;
      }
   }
   private void OnCollisionStay(Collision collision)
   {
      if (KeyInputDelayTimer + 0.5f < Time.time)
      {
         KeyInputDelayTimer = Time.time; // To update the percentage ever 0.5s
         activeTexture = canvasScript.GetPaintMainTexture(paintMaterial); // Get render texture of
             gameobject from material
         thisTexture = new Texture2D(activeTexture.width, activeTexture.height); // Create new
             texture2D using render texture sizes
         RenderTexture.active = activeTexture; // Set as the active texture
         thisTexture.ReadPixels(new Rect(0, 0, activeTexture.width, activeTexture.height), 0, 0); //
             Read rendertexture pixels into texture2D
         pix = thisTexture.GetPixels32(); // Get all the pixels in texture2D into array format in
             RGBA

         for (int i = 0; i < pix.Length; i++) // Compare each pixel's RGB to paint RGB. If within
             range, it's painted on.
         {
            if (InRange(collisionPainterColor.r - 40, collisionPainterColor.r + 40, pix[i].r) &&
               InRange(collisionPainterColor.g - 40, collisionPainterColor.g + 40, pix[i].g) &&
               InRange(collisionPainterColor.b - 40, collisionPainterColor.b + 40, pix[i].b))
            {
               fill++; // Adds +1 per pixel in range
            }
         }
         fillPercent.text = "Fill: " + (fill / pix.Length * 100).ToString("N") + "%"; // Displays
             fill pct into game
         fill2 = fill / pix.Length * 100;
```

```
            fill = 0; // Resets fill
        }
    }

    private void OnCollisionExit(Collision collision)
    {
        if (fill2 >= 95)
        {
            GameControl.instance.gameOver = true; // Gameover
            colorGrabbed = false; // Reset color
            canvasScript.ResetPaint(); // Reset canvas
            fillPercent.text = "Fill: 0.00%"; // Displays fill pct into game
        }
    }

    private bool InRange(int low, int high, int x) // Returns true if x is between the two numbers
    {
        return ((x - low) <= (high - low));
    }
}
```