TOWARDS NATURALISTIC INTERFACES OF VIRTUAL REALITY SYSTEMS

JINGBO ZHAO

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAMME IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

JANUARY 2019

ABSTRACT

Interaction plays a key role in achieving realistic experience in virtual reality (VR). Its realization depends on interpreting the intents of human motions to give inputs to VR systems. Thus, understanding human motion from the computational perspective is essential to the design of naturalistic interfaces for VR.

This dissertation studied three types of human motions, including locomotion (walking), head motion and hand motion in the context of VR.

For locomotion, the dissertation presented a machine learning approach for developing a mechanical repositioning technique based on a 1-D treadmill for interacting with a unique new large-scale projective display, called the Wide-Field Immersive Stereoscopic Environment (WISE). The usability of the proposed approach was assessed through a novel user study that asked participants to pursue a rolling ball at variable speed in a virtual scene. In addition, the dissertation studied the role of stereopsis in avoiding virtual obstacles while walking by asking participants to step over obstacles and gaps under both stereoscopic and non-stereoscopic viewing conditions in VR experiments.

In terms of head motion, the dissertation presented a head gesture interface for interaction in VR that recognizes real-time head gestures on head-mounted displays (HMDs) using Cascaded Hidden Markov Models. Two experiments were conducted to evaluate the proposed approach. The first assessed its offline classification performance while the second estimated the latency of the algorithm to recognize head gestures. The dissertation also

conducted a user study that investigated the effects of visual and control latency on tele-operation of a quadcopter using head motion tracked by a head-mounted display. As part of the study, a method for objectively estimating the end-to-end latency in HMDs was presented.

For hand motion, the dissertation presented an approach that recognizes dynamic hand gestures to implement a hand gesture interface for VR based on a static head gesture recognition algorithm. The proposed algorithm was evaluated offline in terms of its classification performance. A user study was conducted to compare the performance and the usability of the head gesture interface, the hand gesture interface and a conventional gamepad interface for answering Yes/No questions in VR.

Overall, the dissertation has two main contributions towards the improvement of naturalism of interaction in VR systems. Firstly, the interaction techniques presented in the dissertation can be directly integrated into existing VR systems offering more choices for interaction to end users of VR technology. Secondly, the results of the user studies of the presented VR interfaces in the dissertation also serve as guidelines to VR researchers and engineers for designing future VR systems.

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor Prof. Robert Allison, who offered me the opportunity to conduct this work as a PhD student in Canada. This dissertation would not be possible without his guidance and support over the past four years. I had excellent research experience under his guidance, with the freedom to explore the research topics that I was interested in. I also had many opportunities to travel to conferences in very nice places.

I would like to thank my committee members Prof. Petros Faloutsos and Prof. Burton Ma. They have been very supportive and encouraging during my PhD study. I would also like to thank Prof. Jia Xu, Prof. Graham Wakefield and Prof. Andrew Hogue for taking time out of their busy schedule to serve my examination committee.

I would like to thank Prof. James Elder, Prof. John Tsotsos and Prof. Michael Brown - I learnt a lot from you in your classes and seminars. I would like to thank our graduate program director Prof. Simone Pisana and ex-graduate program directors Prof. Franck van Breugel and Prof. Uyen Trang Nguyen for their assistance that enabled me to complete this PhD program. I would like to thank Ms. Ouma Jaipaul-Gill, Ms. Stefanie Caputo, Ms. Susan Cameron and Ms. Teresa Manini for the coordination and the help they offered to me during this entire program. I would like to thank Ms. Ulya Yigit, Mr. Jaspal Singh, Mr. Nam Tran and Ms. Seela Balkissoon for their technical support.

I would like to thank Dr. Margarita Vinnikov and Mr. Sion Jennings for offering me an opportunity to work as a research intern at the Flight Research Laboratory, National Research Council, Ottawa. The work I have done during this internship is now part of the

dissertation (Chapter 6). Thanks to Prof. James Elder for offering me the financial support for the internship.

Finally, I would like to thank my parents for their support. Many thanks go to my lab members, and to the participants who attended my user studies.

PREFACE

This dissertation is based on the following publications:

Paper I **Zhao, J.** and Allison, R. S. (2016). Learning gait parameters for locomotion in virtual reality systems. *2nd International Workshop on Understanding Human Activities Through 3D Sensors*. Lecture Notes in Computer Science, vol 10188, pp 59-73.

Paper II **Zhao, J.,** Allison, R. S., Vinnikov, M. and Jennings, S. (2017). Estimating the motion-to-photon latency in head mounted displays, *2017 IEEE Virtual Reality (VR)*, pp. 313-314.

Paper III **Zhao, J.** and Allison, R. S. (2017). Real-time head gesture recognition on head-mounted displays using cascaded hidden Markov models, *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2361-2366.

Paper IV **Zhao**, **J.,** Allison, R. S., Vinnikov, M. and Jennings, S. (in press). The effects of visual and control latency on piloting a quadcopter using a head-mounted display, accepted by *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*.

Paper V **Zhao**, **J.** and Allison, R. S. (manuscript). Comparing head gesture, hand gesture and gamepad interfaces for answering Yes/No questions in virtual environments.

Paper VI **Zhao, J.** and Allison, R. S. (manuscript). The role of binocular vision in avoiding virtual obstacles while walking.

Related publications not included in this dissertation:

Zhao, **J.,** Bunn, F. E., Perron, J. M., Shen, E. and Allison, R. S. (2015). Gait assessment using the Kinect RGB-D sensor, *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 6679-6683.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

Virtual reality (VR) systems are those that use a combination of trackers, displays, actuators, computational hardware and software algorithms to immerse users into a virtual world and render senses to users based on the contents and interactions in virtual environments. The goal of VR systems is to provide users with immersive and realistic experience to interact with virtual worlds. Four factors in VR systems are sought to be improved by VR researchers and engineers. These include immersion, presence, perception and interaction (Dodiya and Alexandrov, 2007). Among these factors, interaction allows users to interact with virtual worlds in real-time and it is typically realized through the interfaces between users and VR systems.

The naturalism of interfaces is largely dependent on whether the way users interact in virtual environments corresponds to what they do in real worlds. For instance, in the early days of VR research, joysticks were used to walk in VR (Robinett and Holloway, 1992). But now users are able to perform real walking in VR using locomotion interfaces (Hollerbach, 2002) , which make such experience more natural and realistic. Thus, developing naturalistic interfaces is essential to the research and the design of VR systems.

In this dissertation, interfaces in VR systems are categorized into active interfaces and passive interfaces. Active interfaces refer to those devices that are used by users to give inputs to VR systems. These include locomotion (walking) interfaces (Hollerbach, 2002) and hand gesture interfaces (Xu, 2006) and they work by tracking and recognizing

human motion to interpret users' intents to interact with VR systems. Conversely, passive interfaces are the devices that give users feedbacks based on the contents in virtual environments. Typical passive interfaces include displays (Sutherland, 1968; Ware *et al*., 1993; Cruz-Neira *et al*., 1993), haptic interfaces (Choi *et al*., 2016) and olfactory displays (Matsukura *et al*., 2010). Some interfaces can be hybrid. For example, a locomotion interface can be equipped with mechanical structures and actuators to give people sensations during walking. Examples include the feelings of walking on uneven surfaces *(*Noma *et al*., 2000) or climbing stairs (Vu *et al*., 2017). Such integration makes a locomotion interface also a haptic interface. On the other hand, haptic interfaces, such as data gloves, are also motion-tracked, which are able to give inputs to VR systems, so these are also active interfaces.

In general, research on VR interfaces has not been extensively studied and some topics have been ignored. For instance, head gesture is a natural means for communication and interaction between people, but the use of head gesture interfaces to interact with virtual avatars and virtual environments has been largely ignored. Thus, it is desirable to add head gesture interfaces to VR systems for interaction. Similarly, many algorithms have been developed for hand gesture recognition (Cheng *et al.*, 2016), but they have been rarely evaluated for VR interaction. Therefore, interfaces for VR systems, such as head gesture interfaces and hand gesture interfaces, are worth further exploration.

Designing active interfaces for interaction in VR requires understanding of human motion from the computational perspective. The general approach is to capture and analyze

human motion to estimate the motion states using motion capture systems. Estimated motion states usually include the position, the orientation and the velocity of body parts. These motion states can be directly reflected to changes in the user's virtual representation (viewpoint and avatar) to update the user's states in virtual environments. This introduces egocentric motion and motions that can be used for interaction with VR, through means such as gesture recognition. The interaction between users and virtual worlds is calculated by algorithms and the feedbacks from the virtual worlds are rendered through passive interfaces, such as displays and actuators. High-level algorithms may also be developed and used to reveal the metaphor of a sequence of estimated motion states. For example, it is possible to recognize whether users are walking or running in virtual environments based on the trajectories of tracked body parts. Virtual avatars other than the avatars of users are therefore able to understand the motion of users in virtual environments and interact with users. This will be helpful for designing more interactive virtual environments.

Popular motion capture systems for present-day VR research are optical systems and inertial systems, or hybrid systems that are a combination of both. Optical systems use techniques, including triangulation (two or more camera views) (Hartley and Zisserman, 2003), structured light (Shotton *et al.*, 2011) and time-of-flight (Sarbolandi *et al*., 2015), to directly measure positions of tracked body parts. Inertial systems typically use a combination of accelerometers and gyroscopes (Woodman, 2007), mounted on body parts to be tracked, to measure linear acceleration and angular velocity. Hybrid systems also fuse the

data streams from optical systems and inertial systems using algorithms, such as the Kalman filter (Kalman, 1960), to give estimates. Both optical systems and hybrid systems were involved in the experiments presented in the dissertation.

This dissertation studied three types of human motion in VR: locomotion, head motion and hand motion and focused on the design of the corresponding interfaces. The primary goal of the dissertation was to design and evaluate active interfaces in VR, including a locomotion interface, a head gesture interface and a hand gesture interface, based on analyzing captured motion data. The locomotion interface was evaluated through a user study that asked users to pursue a rolling ball in a virtual scene by walking on the locomotion interface while the performance and the usability of the head gesture interface and the hand gesture interface were evaluated and compared with a gamepad interface for answering Yes/No questions in virtual environments. In addition, VR systems provide researchers with a unique way for studying human motion and behaviour as the experiment settings in virtual environments can be well-controlled and motion data can be easily captured in a controlled and tracked lab space. The results of the psychophysical experiments may tell us how people perceive and react and may in turn help to improve VR systems in terms of realism and interoperability. Furthermore, VR also can be used as a tool to model designs in the real-world. Usability studies can be conducted in virtual environments and these studies help to determine whether the proposed design meets the actual requirements. Thus, another goal of the dissertation was to conduct experiments using VR interfaces to observe and investigate locomotion and head motion through two user studies: the first user study

investigated the role of stereopsis in avoiding virtual obstacles while walking in VR and the second study investigated the effects of visual and control latency on piloting a quad-copter using a head-mounted display (HMD); in the second study, a method for objectively estimating the end-to-end latency in HMDs was also presented.

In summary, the main contributions of the dissertation were:

- I presented a locomotion interface based on a Wide Immersive Stereo Environment (WISE) and a treadmill using a machine learning approach. The evaluation of the locomotion interface was conducted through a novel user study that asked users to pursue a rolling ball. The task required users to adjust their walking speed to maintain a distance of 5 m to the rolling ball using the locomotion interface.
- I studied the role of stereopsis in avoiding obstacles and gaps through two walking experiments conducted in VR. Similar to the previous study, these experiments were conducted using the WISE and a linear treadmill. Two virtual scenes were designed. Users were able to perform linear walking in these virtual scenes while avoiding virtual obstacles and gaps.
- I presented a head gesture recognition algorithm that recognized head gestures using a proposed algorithm called the Cascaded Hidden Markov Models (CHMMs). The proposed recognition algorithm can be used as an interface to interact with virtual environments.

- I presented a hand gesture interface that recognized dynamic hand gestures based on a static hand gesture recognition algorithm. I also conducted a user study for the hand gesture interface that asked participants to answer Yes/No questions in virtual environments to compare its performance and utility with the proposed head gesture interface and a conventional gamepad interface.
- I conducted a user study that investigated the effects of visual and control latency on piloting a quadcopter using an HMD. To study whether the end-to-end latency plays a dominant role in the quadcopter control scenario, I also presented a general method for estimating the end-to-end latency in HMDs.

The rest of the thesis is organized as follows: Chapter 2 discusses related work. Chapter 3 presents the design and the evaluation of a locomotion interface. Chapter 4 investigates the role of stereopsis in avoiding virtual obstacles while walking. Chapter 5 presents the designs of a head gesture interface and a hand gesture interface and a user study that compares these two interfaces together with a gamepad interface. Chapter 6 studies the effects of visual and control latency in teleoperation of a head motion controlled quadcopter using an HMD. Chapter 7 draws the conclusion of the dissertation and discusses future work.

Chapter 2
Related Work

2.1 VR Systems and Displays

VR systems integrate different hardware and software components. Typical hardware components include trackers, displays, actuators and computational platforms. These hardware components are usually supported by their own algorithms and software frameworks. For example, trackers may use filtering algorithms to eliminate noise in the tracked human motion signals and graphics cards usually have their rendering frameworks to generate photo-realistic images. As the trackers in VR systems are usually heterogeneous, it is a common practice to use a sensor network (Taylor *et al*., 2001) to support the communication and control across these trackers. On top of these hardware and software components, there is usually a global strategy (or an interaction technique) that defines the behaviours of users and the functions of VR systems. It also circumvents the limitations of hardware and software components.

Displays are indispensable components to VR systems and they are used for viewing virtual environments. The primary types of displays for VR systems include monitors, immersive projection displays (such as cave automatic virtual environments (CAVEs)) and head-mounted displays (HMDs). Ware *et al*. (1993) introduced the Fish Tank VR system. This system used stereoscopic monitors for visualization. Head positions of users were tracked by mechanical sensors and the stereoscopic images of a 3D scene were rendered

based on the tracked head positions. CAVEs (Cruz-Neira *et al*., 1993) are typical immersive stereoscopic displays for VR systems. These displays usually use projectors to project images onto a cube-like structure, which consists of 3-6 wall displays, or onto spherical displays (curved displays). HMDs are devices that are mounted on a user's head for the visualization of 3D scenes. Sutherland (1968) created the first HMD, which integrated head tracking, real-time rendering and a stereoscopic display. Displays are relatively mature technologies but interaction techniques with displays remain an open field and are therefore worth exploring.

2.2 Locomotion

2.2.1 Locomotion Techniques

Locomotion techniques in VR systems include: flying using a joystick (Robinett and Holloway, 1992), leaning (LaViola *et al*., 2001), walking-in-place (WIP) (Slater *et al*., 1995; Templeman *et al*., 1999; LaViola *et al*., 2001; Yan *et al*., 2004; Feasel *et al*., 2008; Bruno *et al*., 2013, 2017; Williams *et al*., 2011; Wilson *et al*., 2014), redirected walking (RDW) (Razzaque et al., 2001, 2002) and numerous mechanical repositioning techniques (Nilsson *et al*., 2013), including treadmills (Souman *et al*., 2008), torus treadmills (Iwata, 1999), virtual perambulators (Iwata, 1999), foot platforms (Iwata *et al*., 2001), pedaling devices (Allison *et al*., 2000) and spheres (Medina *et al*., 2008). For example, flying using a joystick, originally developed for locomotion in CAVEs, enables a participant to locomote by actively operating a joystick with a hand. Later, as locomotion techniques became more sophisticated, more motion cues were considered and included to improve presence in VR

environments. Leaning, for instance, made a step forward in terms of full body control compared to flying by sensing upper body tilt angle and using it for locomotion. The first WIP interface "walking on spot" (Slater *et al*., 1995) used head motion to estimate one's walking speed and more recent WIP techniques (Templeman *et al*., 1999; Yan *et al*., 2004; Feasel *et al*., 2008; Wendt *et al*., 2010; Bruno *et al*., 2013; Wilson *et al*., 2014; Tregillus and Folmer, 2016; Bruno *et al*., 2017) consider human biomechanics by using gait parameters to estimate one's walking speed. These WIP techniques allow a user to step in-place to locomote but only accommodate the motions of stepping not the forward stride of locomotion. The more sophisticated RDW technique makes it possible for a user to perform 'real' walking (e.g., take forward steps) by consistently re-orienting the user toward the center of a constrained space while the user wears an HMD. This circumvents the difficulty that it is often impractical to build a 1:1 virtual scene, for example, when the simulated space exceeds the size of the physical space of the VR display or the VR tracking volume. Another approach to simulate walking while constraining physical motion is mechanical repositioning. These techniques normally use a locomotion interface to cancel the effects of the user's stride, keeping them in-place while performing walking motions. The technique is especially useful for VR systems that use large-scale projected displays for visualization. Walking biomechanics in these techniques are usually monitored by optical trackers, inertial trackers or hybrid trackers mounted on lower limbs. In summary, the general goal for designing a locomotion technique is to make it as natural as real-walking.

Introducing bipedal walking in VR elicits proprioceptive cues that make walking experience in VR more naturalistic (Langbehn *et al*., 2018). In addition to proprioceptive cues, vestibular cues are also important for natural experience in VR, which can be provided by RDW techniques (Langbehn *et al*., 2018).

Researchers have also evaluated and compared different locomotion interfaces. Nabiyouni *et al*. (2015) evaluated the Virtusphere technique, the real walking interface and the gamepad interface. They showed that the Virtusphere as a moderate-fidelity technique was significantly outperformed by a high-fidelity real walking interface and a well-designed low-fidelity gamepad interface as the Virtusphere was fatiguing and difficult to control due to its large inertia. Conversely, the real-walking interface was natural to people and the gamepad interface had a clear mapping between joystick movement and users' intended direction of travel so it was easy to use. Kitson *et al*. (2017) compared several seated leaning locomotion techniques using different types of chairs to the joystick interface. They reported that participants in general preferred the leaning techniques as they are fun, engaging and more realistic but the joystick interface was still easier to use and control. Zielasko *et al*. (2016) evaluated five locomotion techniques. Among these techniques, the Adapted Walking in Place and the Accelerator Pedal involved lower limb movements. Leaning required upper body movements while seated. The Shake Your Head technique used only head movements tracked by an HMD. These techniques were also compared with the traditional gamepad interface. They found that the Accelerator Pedal and the Leaning technique performed better than other techniques in terms of user preference and task

performance. Most recently, Coomer *et al*. (2018) compared four locomotion methods, including the joystick interface, the Arm-Cycling, the Point-Tugging and teleporting. The Arm-Cycling is a locomotion technique that creates egocentric motion in VR based on the displacements of HTC Vive controllers held in users' hands when users perform cycling motion of their arms with the triggers on the HTC Vive being pressed down. The Point-Tugging is method that requires users to grab a virtual point in virtual environments by pressing the triggers on the HTC Vive controllers and then tug to move themselves in virtual environments, followed by releasing the triggers to complete the movement. They concluded that the Arm-Cycling was the best locomotion method among these four techniques as it gave better sense of spatial awareness and lower simulator sickness scores.

Although many locomotion interfaces have been developed for mechanical repositioning in VR systems, 1-D treadmills are still of interest since they can be easily obtained. Many studies have focused on developing methods for automatic speed adaptation of 1-D treadmills and these methods can be generally divided into four categories (Park *et al*., 2015): inertial-force-based controllers, position-based controllers, physiology-based controllers and gait-parameter-based controllers. For example, von Zitzewitz *et al*. (2007) developed a force-based controller by which a participant is required to wear a mechanical harness equipped with force sensors to measure the ground reaction force between their feet and the belt of a treadmill. The control of the acceleration of the treadmill is determined based on Newtonian mechanics using the measured ground reaction force. Souman *et al*. (2010) developed a position-based controller in which the walking speed is estimated from

the deviation of the position of the user from a reference position on the treadmill. The physiology-based controller by Su *et al*. (2005) used heart rate in a biofeedback mode to control the speed and the slope of a treadmill to help a user maintain heart rate. Yoon *et al*. (2012) developed a combined gait-parameter-based and position-based controller, which estimates a user's walking speed from the peaks of foot swing velocity and the position of a user on the treadmill. Wiens *et al*. (2017) developed a similar method to the approach by Yoon *et al.*, which was to adjust the speed of treadmill based on user's position on a treadmill and leg swing velocity. Generally, the goals of such controllers are to keep a user on the treadmill by estimating a user's intended walking speed and adapting the speed of the treadmill using the speed estimate. The related parameters that can be used for such estimation include ground reaction force, the user's position on treadmill, physiological factors or gait parameters, *etc*. However, estimation of walking speed using machine learning techniques based on gait parameters has not been explored. Thus, it is a promising research topic.

2.2.2 The Role of Stereopsis in Walking

Locomotion techniques in VR allow researchers to study human locomotion behaviors as it is relatively easy to set up experimental scenes and perform motion tracking in VR than in the real-world. The role of stereopsis in walking has been studied in small physical space in laboratories but has not been studied in a virtual environment that simulates a much longer walking distance. Thus, it is a potential research topic that can be studied with VR locomotion techniques.

Previous research has shown that stereopsis aids activities related to hand-eye coordination (Fielder and Moseley, 1996), but it is less clear that stereopsis provides advantages in locomotion activities, such as walking and running, as steady viewing is needed to let stereopsis achieve maximum precision (McKee *et al*., 1990). Some studies have shown that stereopsis also helps people to make more accurate lower limb movements (Patla *et al*., 2002; Loomis *et al*., 2006; Hayhoe *et al*., 2009; Chapman *et al*., 2012).

For example, Patla *et al*. (2002) conducted an experiment to study the role of stereopsis in locomotion by asking participants to step over a single obstacle in a straight path. Their finding was that toe clearance was increased under non-stereoscopic viewing compared to stereoscopic viewing, which indicated that stereoscopic viewing improved lower-limb lift accuracy or that people acted more cautiously in absence of stereoscopic viewing. Loomis *et al*. (2006) had participants to go through a small field with randomly placed obstacles to reach a goal at the other end of the field, while avoiding collision with the obstacles. Results showed that stereoscopic viewing resulted in fewer collisions compared to non-stereoscopic viewing than non-stereoscopic viewing. Hayhoe *et al*. (2009) studied the role of stereopsis in locomotion by asking participants to walk in an indoor environment with two obstacles and one table. The task was to step over two given obstacles, go around the table and step over the two obstacles again before returning to the start point. They found that stereoscopic viewing gave shorter task completion time and lowered foot clearance height (toe clearance). The finding on the foot clearance height was consistent with that of Patla *et al*. (2002). Chapman *et al*. (2012) investigated the influence of stereopsis in

foot placement accuracy using a task that asked participants to walk in a straight path and step on floor targets as accurately as possible. Each floor target consisted of two pieces of white tape angled 90 degrees to form to a corner of a square. They found that, under non-stereoscopic viewing, foot placement was less accurate in medio-lateral plane and terminal foot-reach duration was longer compared to that of stereoscopic viewing.

A limitation of these studies was that the experiments were conducted in setups with limited walking distances that did not represent typical walking scenarios in our everyday life - we usually walk continuously over longer distances. Thus, it is still uncertain whether stereopsis helps people to make more accurate movements under constant motion during continuous walking. VR systems provide us with a unique opportunity for simulating open and large environments, they are promising platforms to investigate the influence of stereopsis on gait parameters.

2.3 Head Motion

Tracking head motion is important and ubiquitous for VR. Firstly, images of VR environment presented in VR displays are rendered based on the tracked head position in the physical space. Secondly, tracking head motion allows interaction in VR. For example, it is possible to implement locomotion in VR by using head motion alone (Slater *et al*., 1995; Tregillus and Folmer, 2016) and it is also possible to implement a head gesture interface based on head movements for interaction, which is presented in Chapter 5. Thirdly, head motion tracked by an HMD enables us to teleoperate vehicles. The surroundings of vehicles can be observed on the HMD. This usually gives an egocentric, immersive and intuitive

way of control than the conventional way that normally uses a joystick and a monitor. In addition, the usability and performance of teleoperation of vehicles using an HMD can be directly studied with VR experiments.

2.3.1 End-to-End Latency

Head motion in VR systems is typically coupled with updates of visual contents on displays. Changes in head position and orientation in tracked physical space must be immediately updated in virtual environments and presented onto displays. A critical parameter involved in the process is the end-to-end latency, which is defined as the time interval between a user's physical motion and the resulting update of a new frame presented on the display due to the motion. The parameter has been recently referred to as the motion-to-photon latency (Iribe, 2013). Optimal VR experience requires the latency to be less than 20 ms, and 60 ms is usually taken as an upper bound for an acceptable VR experience (LaValle *et al*., 2014), depending on the task and the nature of head movements. High latency can result in instability of the environment, cause simulator sickness and affect task performance (Allison *et al*., 2001). To measure the latency of VR systems, objective methods are needed. Many approaches have been proposed to estimate the end-to-end latency in CAVEs (Steed, 2008) and HMDs (Di Luca, 2010; Raaen and Kjellmo, 2015; Kijima and Miyajima, 2016; Seo *et al*., 2017). Such methods usually use mechanical structures to introduce motion to the tracker of the display, video cameras and photodiodes are used to monitor the motion of the tracker and the changes on the display. In addition, the motion of the tracker also can be monitored by a rotary encoder. Most recently, to estimate the latency of the Oculus

Rift DK2, Kijima and Miyajima (2016) stacked two high-speed cameras on a turntable rotating back and forth according to a semi-triangle wave. One camera captured a head tracked vertical white line rendered on the HMD while the other captured a static vertical white line in the real world. The latency was estimated by performing cross-correlation on the angular directions of both the virtual and physical white lines. Similarly, Raaen and Kjellmo (2015) compared light sensor outputs monitoring the position of a laser pointer mounted to an HMD with another sensor monitoring changes in brightness on the HMD. However, these studies did not evaluate the latency of an HMD given different initial accelerations. In addition, the comparison between the translational latency and the rotational latency of an HMD has not been done. Thus, the dissertation provides an alternative approach for estimating the latency of HMDs, use it to estimate the latency given different initial accelerations and compare the translational latency and the rotational latency.

To minimize the motion-to-photon latency, it is possible to predict head motions assuming constant head rotational speed or acceleration (LaValle *et al*., 2014). Measurement of VR systems with prediction algorithms is more challenging as the motion-to-photon latency is a variable parameter due to prediction algorithms and the details of implementation of the prediction algorithms are usually proprietary and are not available for examining.

2.3.2 Head Gesture

Head gesture is a natural means of face-to-face communication between people but the recognition of head gestures in the context of virtual reality and use of head gesture as an

interface for interacting with virtual avatars and virtual environments have been rarely investigated. Recent advances in motion tracking technologies have enabled users' head movements to be accurately tracked in real-time. For example, HMDs, such as the Oculus DK2, integrate head motion tracking sensors (LaValle *et al*., 2014). Large projective displays, such as the CAVEs, are usually equipped with head tracking glasses. These devices make real-time head gesture recognition possible in VR environments.

Recognizing gestures is usually considered as the problem of recognizing sequences and Hidden Markov Models (HMMs) (Rabiner, 1989) have been widely used for recognizing hand and body gestures (Campbell *et al*., 1996; Hossain and Jenkin, 2005; Chen *et al*., 2009; Liu *et al*., 2014). Previous studies on head gesture recognition were mostly computer vision based systems using HMMs (Morimoto *et al*., 1996; Terven *et al*., 2014). In such systems, a user's face was usually captured by a camera and computer vision algorithms were used to track a user's face and estimate the user's head orientation from the tracked face.

Morimoto *et al*. (1996) used an optical flow algorithm to estimate the yaw, pitch and roll of a user's head. The estimated angles were quantified into seven observation symbols by thresholding. The thresholds were determined by calculating the energy of a sequence of angles, but the details of the energy calculation algorithm were not described. The observation symbols were used to train four HMMs. These HMMs correspond to four gestures: Yes, No, Maybe and Hello. Three gestures Yes, No and Maybe were modeled with fully connected HMMs while the gesture Hello was modeled with a left-right HMM.

The classification performance was evaluated offline using sequences of gestures collected from several participants. Terven *et al*. (2014) used the Supervised Descent Method (SDM) to extract 2-D facial features. The yaw, pitch and roll were estimated by using the Pose from Orthography and Scaling with Iterations (POSIT) method with the extracted 2-D facial features and a 3-D anthropometric head model. The symbols were generated by comparing changes in yaw and pitch across consecutive frames. To cover six different head gestures: Nodding, Shaking, Left, Right, Up and Down, the researchers trained six HMMs and these models were fully connected to form a cyclic structure. To evaluate the classification performance, videos that contained different head gestures were recorded and the performance was evaluated offline.

Although computer vision based systems using HMMs for head gesture recognition exist, to my knowledge, HMMs have not been used for recognizing head gestures on HMDs. The performance and usability of head gesture interface in virtual environments have not been studied. Thus, the dissertation presents a method for recognizing head gestures on HMDs and conducts a user study to evaluate its performance and usability in virtual environments.

2.3.3 Teleoperation of Vehicles and Onboard Cameras through Head Motion and Its Effects of Latency

Head motion is useful for controlling vehicles and onboard cameras using HMDs. Earlier research proposed to jointly use an HMD and a joystick to respectively control the onboard camera and the vehicle. For instance, de Vries and Padmos (1997) compared different

viewing conditions for operating a camera on a simulated unmanned aerial vehicle (UAV) using head motions tracked by an HMD while piloting the UAV with a joystick. Morphew *et al*. (2004) studied the performance of two methods (a joystick with an HMD *vs* a joystick with a computer monitor) for controlling a UAV. Their study found that using a joystick with a computer monitor gave better task performance than using a joystick with an HMD and elicited less simulator sickness. Mollet and Chellali (2008) proposed the idea of using head tracking functionality in HMDs for teleoperation of cameras on robots to improve the degree of immersion by presenting the view from the robot vantage point. Their prototypes were based on wheeled-robots and head rotation was used to control the attitude of an onboard camera. But the evaluations of the proposed system were not given in the paper. Doisy *et al*. (2017) compared the performance of three methods for controlling robots. Experiment conditions included: (a) an Xbox 360 controller for controlling both robot movement and camera orientation. (b) an Xbox 360 controller for controlling robot movement and an HMD for controlling camera orientation. (c) hand gestures for controlling robot movement and an HMD for controlling camera orientation. Similar to the results by Morphew *et al*. (2004), they found that using the Xbox 360 controller for controlling robot movements and camera orientation had the best task performance. A more recent work by Smolyanskiy and Gonzalez-Franco (2017) presented a design of a quadcopter with stereoscopic viewing using the Oculus Rift DK2. Onboard camera panning was controlled digitally (as opposed to mechanically) by head motion while the quadcopter was piloted by a hand-held controller. Real flight tests were carried out to study the simulator sickness with

an end-to-end latency of 250 ± 30 ms and participants had minor simulator sickness. Another experiment was conducted to study simulator sickness in relation to gaming experience and visual acuity using pre-recorded flight videos.

More recently, systems that teleoperate vehicles or robots using head motion tracked by an HMD have been developed. In these systems, egocentric views captured from the onboard cameras from the vehicles or robots are presented onto the HMD. Such settings give users an egocentric, immersive and intuitive way of teleoperation compared with conventional control methods using hand-held devices. Several projects have explored teleoperation of robots and quadcopters using an HMD: Martins and Ventura (2009) presented a head motion controlled search and rescue (SAR) robot using an HMD. The HMD in the design had a three Degree-of-Freedom (DOF) tracker, which could capture yaw, pitch and roll angles of a user's head. The SAR robot had two tracks and an articulated frontal body with a stereo camera mounted on it. The yaw angle of a user's head was mapped to the angular velocity of the rotation of the SAR robot and the pitch angle was directly mapped to the angle of the front body of the robot to ascend or lower it. Since the robot was unable to perform roll movements, the roll angle of a user's head was used to rotate the image presented on the HMD. The experimental results showed that the control that utilized the HMD gave better performance in terms of depth perception, detail perception and the execution of a SAR operation compared to a 2D interface on a computer monitor with joystick control. Higuchi *et al*. (2013) proposed a control mechanism for quad-

copters called the Flying Head. In this method, translation motion accomplished by walking and tracked by an HMD was directly mapped to the spatial motion of a quadcopter i.e. the *x*-, *y*- and *z*-positions and the yaw angle were synchronized between the user and the quadcopter. Experimental results showed that the proposed control method outperformed the conventional control method using a joystick when tracking static or moving targets. Pittman and LaViola (2014) evaluated six different techniques for operating a quadcopter, including five techniques based on head motion and body motion using an HMD and a technique using the Wiimote. To evaluate all six techniques, participants were asked to fly a quadcopter through five archways placed in an open environment. However, their results showed the Wiimote technique led to the shortest task completion time. Users had lower simulator sickness scores using the Wiimote interface compared with other interfaces. We note that in first five techniques, the control mechanisms were discrete ON/OFF control inputs. This may introduce higher latency and increase discrepancy between the intended motion, the vestibular cues and the visual cues. Thus, it is not surprising that performance of participants using the five head motion control techniques was lower compared to the Wiimote interface and the degree of simulator sickness was higher. Teixeira *et al*. (2014) used an augmented reality (AR) device - the Google Glasses for controlling a quadcopter. In this work, the gestures of the Google Glasses were mapped to the motion of the quadcopter and the video stream captured by the onboard camera of the drone was presented to the right eye of users. An advantage of the design is that operators know both their egocen-

tric positions and the position of the quadcopter while operating the quadcopter. Thus, operators may simultaneously use other devices while flying the quadcopter. But the design may also confuse operators since visual inputs came from two different sources, which may potentially cause simulator sickness. Some preliminary evaluations regarding the interaction between operators and drones were conducted. But no details were given.

In such systems, a typically overlooked factor is the introduced latency, which included visual latency and control latency. Visual latency (Jennings *et al*., 2004; Blissing *et al*., 2016) refers to the time delay between when an image is captured by a camera and when the photons of the captured image are emitted by the display. Control latency (Jennings *et al*., 2004), on the other hand, is the time delay between when a control command is given to a vehicle and when the vehicle is actuated by the control command. Visual latency and control latency may result in simulator sickness and researchers have investigated the effects of latency on controlling vehicles, including cars and helicopters while the operators were in these vehicles. Blissing *et al*. (2016) investigated the effects of visual latency on driving performance through an experiment conducted on real cars. Participants wore see-through HMDs modified from an Oculus Rift DK2 while driving. End-to-end latency was injected by delaying captured real-word video frames. The task was to drive a car through a slalom course under three different latency levels. Results showed that participants could compensate for increased latency by adapting their driving behaviors. But the tests were short-duration so simulator sickness would not be expected and it was not

assessed. Jennings *et al*. (2004) studied the relationship between latency and flight performance of helicopter pilots. Two experiments were conducted to evaluate visual latency and control latency on piloting helicopters. These included asking pilots to fly a helicopter through two designated courses while wearing HMDs and perform precision hovering in a flight simulator with a motion base. Results showed that both visual latency and control latency degraded flight performance. The frequency and the intensity of simulator sickness increased as longer delays were given. It is important to note that the settings of these experiments were different from the teleoperation of vehicles where operators are not in the vehicles. In the latter case, there is no vestibular cue, which causes a conflict between visual and vestibular systems so simulator sickness can be expected.

In summary, teleoperating a vehicle with an HMD and a joystick does not provide performance advantages compared with teleoperating a vehicle with a computer monitor and a joystick. The former (de Vries and Padmos, 1997; Morphew *et al*., 2004; Mollet and Chellali, 2008; Smolyanskiy and Gonzalez-Franco, 2017; Doisy *et al*., 2017) usually elicits symptoms of simulator sickness. However, operating vehicles using only HMDs (Martins and Ventura, 2009; Higuchi *et al*., 2013; Teixeira *et al*., 2014; Pittman and LaViola, 2014) seem to give people a more direct, immersive and intuitive way of control than joint use of computer monitors and joysticks. In addition, researchers have also investigated the effects of latency on driving cars (Blissing *et al*., 2016) and piloting helicopters (Jennings *et al*., 2004). However, no research has been conducted to study the effects of various levels of latency of head motion controlled quadcopters or other types of vehicles. The latency in

the head motion controlled quadcopters differs with that of the helicopters. In helicopter control, the visual updates are coupled with the hand motions of pilots. Latency in such cases can be divided into visual latency and control latency. In head motion controlled quadcopters, visual updates and quadcopter motions are coupled with head motions and thus visual latency and control latency are combined into a single factor. Since using head motion tracked by an HMD to teleoperate a quadcopter or other robotic platforms is a very promising and popular interface, the effects of visual and control latency in such scenarios are worth direct investigation.

2.4 Hand Motion

Traditionally, a way to give inputs to VR systems is to use hand motion tracked by mechanical hand-held devices, such as joysticks, gamepads and data gloves. This enables users to perform various activities. For example, as mentioned in Section 2.2, flying using a joystick (Robinett and Holloway, 1992) was the first method to locomote or navigate in virtual environments. A problem using hand-held devices in VR is that it prevents users from using their hands to directly perform other natural activities such as object picking and manipulation or making hand gestures to communicate with other people or virtual avatars in virtual environments. Thus, it is beneficial to free users' hands from operating hand-held devices and integrate a hand gesture interface to VR systems so that users can directly use hand gestures to perform activities in virtual environments.

Depth sensors, such as the Leap Motion, have recently become the state-of-the-art devices for tracking hand movements. With datasets collected by these sensors, a large

volume of algorithms that can recognize hand gestures have been developed (as comprehensively reviewed by Cheng *et al*. (2016)). However, these algorithms were primarily evaluated on pre-collected datasets offline. The usability and performance of the hand gesture interfaces for interaction with VR environments in real-time have not been explored. In this dissertation, a dynamic hand gesture recognition algorithm is presented as a hand gesture interface and its performance and usability are compared to the proposed head gesture interface and a gamepad interface through a user study.

Chapter 3
Locomotion in Virtual Reality based on Gait Parameters

3.1 Introduction

In this study, I was interested in developing a combined gait-parameter-based and position-based controller for controlling a 1-D treadmill to walk in VR. The first motivation for this approach was that gait parameters contain explicit cues for estimating a user's intended walking speed. However, the proposed gait-parameter-based controller only estimates the intended walking speed of a user without considering the user's position on the treadmill. To account and control for position, I introduced a position-based controller that estimates a user's position on the treadmill from the user's captured foot position data. This position estimate is used to adjust the position of a user on the treadmill. Other parameters, such as the ground reaction force and the physiological factors could also be included to adapt the current design for future research in specific domains, such as rehabilitation. Secondly, while recent WIP techniques have used gait parameters for walking speed estimation, to my knowledge, the use of gait parameters for mechanical repositioning techniques has not been thoroughly investigated and offers potential for improved interaction, as has been demonstrated by recent WIP techniques. Thirdly, the method is novel in that it uses a machine learning approach to map the defined gait parameters (gait features) into a user's intended walking speed and such techniques have not been investigated for real-time treadmill locomotion interfaces. The work done by Park *et al*. (2012) is related to my approach

but is intended for indoor over-ground walking speed estimation rather than control. They used features that consisted of gravity components and DFT components extracted from the acceleration data collected by a handheld device and trained regularized kernel methods to estimate a user's walking speed. Their approach estimated a user's average walking speed from data sequences collected over several seconds (2.56 or 5.12 seconds for over-ground walking). I instead aimed to estimate a user's intended walking speed within 0.53 seconds (given a sampling frequency of 120 Hz and a queue of a length 64 for buffering foot position data) for real-time control of a 1-D treadmill using low-latency gait features extracted from the foot motion data collected by an optical motion capture system. The current study demonstrates that it is possible to achieve the goal of an interactive real-time locomotion interface using a machine learning approach. Finally, while the techniques described in the study are general and could be applied to a variety of HMD or projective VR display setups, I developed and evaluated the system in the context of a novel immersive projected VR display. This unique new display is known as the Wide-Field Immersive Stereoscopic Environment (WISE). It has been set up at York University to explore locomotion techniques and other interaction techniques with large-scale projective displays and investigate human locomotion and navigation behaviors in controlled VR environments.

In the current study, I present a machine learning approach for developing a locomotion technique based on a conventional 1-D treadmill for interacting with the display. In addition, previous studies (Su *et al*., 2005; von Zitzewitz *et al*., 2007; Souman *et al*., 2010) evaluated their speed estimation algorithms by asking users to walk at their preferred

speed on the treadmill. These evaluations were performed in terms of the estimated walking speed pattern within one gait cycle (von Zitzewitz *et al*., 2007), gains of control law (Souman *et al*., 2010) and the error of walking speed estimation (Su *et al*., 2005). Von Zitzewitz *et al*. (2007) also used a questionnaire to let participants rate the performance of their proposed method. However, these studies did not engage users in an active VR environment during evaluation. To assess my locomotion technique in a more interactive and relevant way, I present a novel user study for assessing the utility of locomotion interfaces by asking a user to pursue a rolling ball in a virtual scene requiring them to control their speed and relative position. While constrained to meet needs for experimental control, such a task is representative of a broad range of navigation tasks in VR.

3.2 Method

I use a set of spatial-temporal features that can be extracted in real-time during actual locomotion with low latency. Inspired by Yoon *et al*. (2012), I select the peaks of foot speed curves as one of the features for estimating intended walking speed. But contrary to their approach that models the foot speed curves as sinusoidal waves using the peaks as the amplitudes and that estimates the walking speed by calculating the average amplitude of the sinusoidal waves and by further fitting the estimated walking speed to the real walking speed using quadratic regression, my assumption is that when walking at a specific speed either over-ground or on the treadmill, the peaks of a foot speed curve should fall into a certain interval. Thus, conversely, if I can estimate the amplitude of the peaks of a foot speed curve, it should be possible to estimate the corresponding intended walking speed.

The second feature that I use is the time intervals between consecutive peaks since the time intervals are inversely proportional to step frequency. Thus, for every detected peak, the algorithm generates a 2-D feature vector. With enough data on foot speed collected from several participants, the problem of intended walking speed estimation can be turned into a classification task by learning the extracted features and using the learning result to estimate a user's intended walking speed. In my implementation, the classification is performed by a typical machine learning approach: the k-nearest neighbor (KNN) algorithm due to its simplicity for multi-class classification. Other machine learning algorithms such as the Support Vector Machine and neural network algorithms may also be used, and further investigation is needed to compare their performance with the KNN-based approach. The result of the classification is the user's intended walking speed. The proposed technique handles two cases of locomotion behavior on the treadmill using a single workflow. First, when a user maintains current walking speed, the extracted 2-D feature vector will consistently fall into a specific interval. The estimated intended walking speed generated by the classification algorithm will be a constant speed value that maintains the speed of the treadmill. Second, when a user intends to change walking speed by increasing or decreasing foot swing speed, the extracted 2-D feature vector will jump from their previous interval to a new interval and the classification algorithm will generate an updated speed estimate that adapts the speed of the treadmill to accommodate the changed motion of the user. If the user maintains the new walking speed, the user's locomotion behavior returns to the

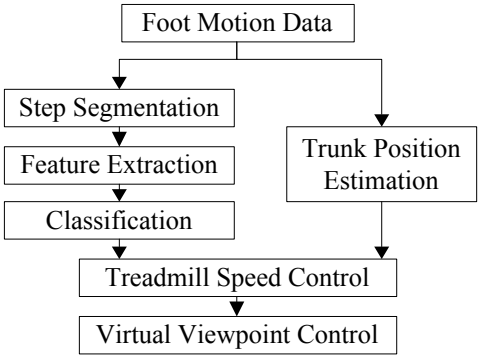


Figure 3-1: Overview of the Proposed Approach

first case and the algorithm will generate constant speed estimates again. The second difference of my work compared with that of Yoon *et al*. (2012) is that I only use foot position data to estimate a user's position on the treadmill, while their method directly captures the position of a user from an infrared (IR) marker attached on the waist. My assumption is that the positions of two feet are symmetrical to the trunk in healthy gait. Thus, the average of the positions of two feet can be considered as the estimate of the position of the trunk.

With the user's intended walking speed from classification and the position information estimated from a user's foot motion, the speed command for controlling the treadmill can be obtained. The actual running speed of the treadmill is queried periodically to update the speed of the virtual viewpoint in a scene. Figure 3-1 presents an overview of the proposed approach.

3.2.1 Hardware and Software of the VR System

The sensor for capturing foot motion is the NaturalPoint OptiTrack V120:Trio, which has three cameras with a resolution of 640 × 480 and a frame rate of 120 Hz. The sensor is an

active IR camera that emits IR light, which is reflected by IR markers and captured by the sensor. To capture human motion, markers are attached to body parts to be tracked. The positions of the markers are calculated by the system's supporting software (Motive 3.7) and broadcast through a network to be shared with other application software. The 1-D treadmill for the study is a commercial type LifeSpan TR5000-DT5. The size of the top surface of the treadmill belt was 51 cm (W) × 142 cm (L). The control of the treadmill is performed through Universal Asynchronous Receiver/Transmitter (UART) controllers by a host machine using a baudrate of 4800 bit/s. The virtual environment is presented on the Wide-Field Immersive Stereoscopic Environment (WISE), which is a curved and large-scale projective display designed by Christie. The images rendered on the display are cast and seamlessly merged by eight overlapping projectors with blending and luminance calibration performed in hardware. Each projector is driven by a client machine (HP Z820 Workstation with Nvidia Quadro k5000 graphics card) in a real-time rendering cluster. The rendering and the synchronization between the host machine (HP Z820 Workstation with Nvidia Quadro k5000 graphics card) and the client machines are handled by the VR software Worldviz Vizard 5.0. The proposed approach was implemented using Python 2.7.

3.2.2 Data Collection and Buffering

To perform step segmentation and detect the peaks of foot speed, it is necessary to buffer a certain amount of data. I use three queues for buffering the timestamps of frames and the z-positions (in depth) of the left foot and the right foot, respectively. The sizes of the queues

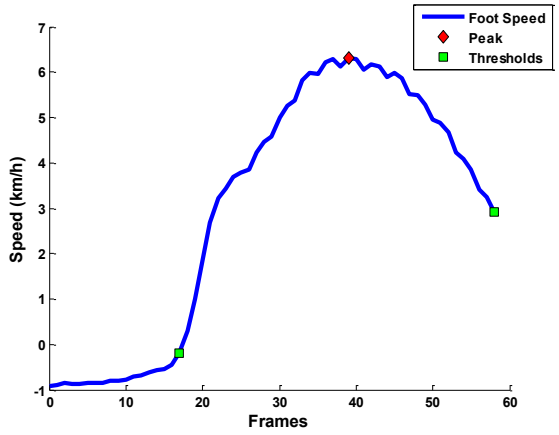


Figure 3-2: Step Segmentation Example

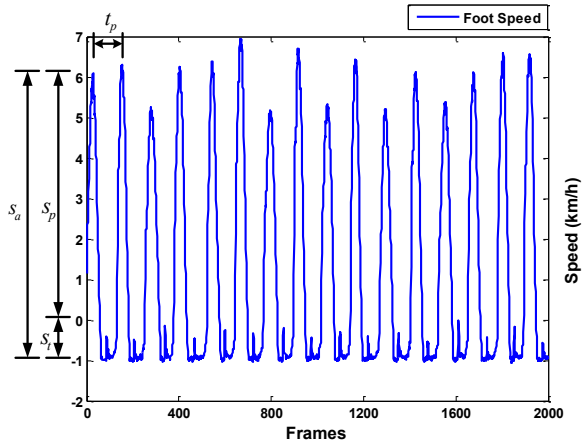


Figure 3-3: Definition of the Gait Features

are chosen to be 64, which gives an initial latency of about 533.3 ms for filling the buffers when the capture starts. The sizes of the queues ensure a complete step can be captured and that the initial latency for buffering is relatively small.

3.2.3 Step Segmentation and Feature Extraction

I use the foot speed for step segmentation because foot speed generally shows better regularities in relation to the phases of a gait cycle than foot position does by manually examining the captured data traces of foot speed and foot position. Foot speeds are calculated

using the buffered position data and the calculated speeds of the left foot and the right foot are then merged together by taking the maximum of the data elements of the left foot and right foot with the same index, since it is not necessary to distinguish whether a detected step is from the left foot or the right foot when estimating walking speed. A gait cycle has three important phases: the initial swing, the mid swing and the terminal swing (Gamble and Rose, 1994). In terms of foot speed, these phases correspond to the moments when a foot gains initial speed due to lifting; the foot reaches the maximum speed in mid-air; and the foot decelerates to a very slow speed before touching the ground, respectively. Theoretically, successful step segmentation involves the detection of these three phases. My dynamic thresholding technique works by first locating a step with a high threshold. From the thresholded data points of a step, the algorithm uses a gradient descent strategy to locate the data points for the initial swing and the terminal swing. The gradient descent algorithm stops whenever it reaches the minimum speed threshold for step segmentation or it detects that the gradient is ascending, indicating a different step is detected. Once the initial swing and the terminal swing are successfully located, I select the maximum speed value in the interval between these two data points and consider it as the data point for the mid swing. Use of the gradient strategy rather than using thresholding directly avoid incorrectly detecting small noisy peaks as steps. However, during actual step segmentation, accurate detection of the terminal swing is not necessary. This is because the algorithm can detect the critical mid swing point before a user completes an entire step. Figure 3-2 shows the scenario, in which the initial and mid swing phases are detected while the terminal swing has

not been buffered. The algorithm stops at the last data point which is sufficient to bracket the mid swing point. In this way, I can ensure timely detection of the mid swing and adaptation of the speed of the treadmill before the swing foot touches the treadmill belt. This minimizes the error between the user's trunk position and the reference position on the treadmill. The step segmentation algorithm is executed whenever a new data frame comes into respective buffers. Thus, the same peak is repetitively detected. But each peak is used for adapting the speed of the treadmill only once. The algorithm was simple with low overhead.

The peak of the mid-swing during treadmill walking is not the same as the peak during over-ground walking. A major difference between over-ground walking and treadmill walking is that the peak value detected during treadmill walking is shifted downward (backwards motion) by the speed of the treadmill belt (Yoon *et al*., 2012) as the foot swing velocity is relative to the speed of the treadmill belt. Thus, whenever a peak is detected, it is necessary to add the current speed of the treadmill belt s_t to the detected peak s_p to yield the real amplitude of the peak s_a. This gives:

$$s_a = s_t + s_p \tag{3.1}$$

The second feature, the time intervals between consecutive peaks t_p, are calculated by subtracting the timestamps of adjacent detected peaks. The initial latency for measuring the time interval is approximately one and a half steps, as the algorithm needs to obtain the timestamps of two successive peaks. Figure 3-3 shows a foot speed curve when the treadmill runs at 1 km/h and gives an illustration of the defined gait features. These two features

constitute a 2-D feature vector: $x = (s_a,\ t_p)$, which can be used for training and testing in the classification step.

3.2.4 Classification

I divided the collected data from participants into halves: one for training and the other for testing the classifier. The class labels of extracted feature vectors y were their corresponding treadmill speeds and satisfy $y \in \{1km/h, 2km/h, 3km/h, 4km/h, 5km/h\}$. Since the features were in different units: km/h for amplitudes of peaks s_a and seconds for time intervals t_p, to avoid biasing the result of classification due to the unit mismatch; I calculated the minimum and maximum values for both features in the training set and performed a linear transformation on all features in the training set to map them into the interval [0, 1] using min-max normalization. For the testing set, I used the minimum and maximum values calculated from the training set and mapped the features in the testing set into the interval [0, 1]. For classification, given a feature vector x, I wished to predict its class label y using a KD-tree version of KNN algorithm (Bentley, 1975). The Euclidean distance was used as the metric for similarity between feature vectors. During the testing phase, first a KD-tree was built using the training set. Then, for each feature vector x in the testing set, the algorithm found its three nearest neighbors. The mode of the class labels y of these neighbors was considered as the output: the estimated walking speed $\hat{s}_w$.

3.2.5 Control of Treadmill Speed and Virtual Viewpoint

The estimated walking speed $\hat{s}_w$ does not account for the user's position on the treadmill. To adjust the position of a user on a treadmill, I estimate a user's trunk position on the treadmill through tracked foot position data. Specifically, I calculate the average foot positions $\overline{p}_f$ and consider it as the estimated position of the trunk $\hat{p}_t$:

$$\hat{p}_t = \overline{p}_f = \frac{\sum_{i=1}^{w_{lf}} p_{lf}^i + \sum_{i=1}^{w_{rf}} p_{rf}^i}{w_{lf} + w_{rf}} \tag{3.2}$$

where w_{lf} and w_{rf} denote the sizes of the queues for buffering the left foot position p_{lf} and the right foot position p_{rf}. In my case, both values equal 64. The speed value s_t' given to the treadmill for control is defined as:

$$s_t' = \left(\hat{p}_t - p_{ref}\right) \cdot k_p + \hat{s}_w \tag{3.3}$$

where p_{ref} is a reference position (typically the center) on the treadmill relative to the IR sensor and k_p is the positional gain. The positional term $\left(\hat{p}_t - p_{ref}\right) \cdot k_p$ compensates for small errors between the position of the trunk and the reference position on the treadmill while the gait-parameter term $\hat{s}_w$ accommodates speed changes.

The virtual viewpoint in a scene is controlled by querying the current speed of the treadmill s_t and using it to update the speed of the viewpoint. The frequency for querying speed is 5 Hz.

3.3 Experiment 3-1: Training and Testing of the Speed Estimation Algorithm

3.3.1 Introduction

The purpose of Experiment 3-1 was to collect the foot motion data to build the training and testing sets for the KNN algorithm and to evaluate the classification performance of the proposed approach. The trained classifier resulting from Experiment 3-1 was also used for the locomotion interface evaluated in the user study in Experiment 3-2.

3.3.2 Participants

Eleven people (3 males, 8 females, age: 21 - 30) participated in the study. Informed consent was obtained from all participants in accordance with a protocol approved by the Human Participants Review Subcommittee at York University.

3.3.3 Procedure

In Experiment 3-1, the IR sensor was placed approximately 1 m in front of the treadmill and participants attached an IR marker on the toe cap of each shoe and a third IR marker on the trunk. Each participant performed two sessions of treadmill walking. In each session, the treadmill started 5 seconds after the experiment began. Data collection was started at the same time. The speed of treadmill started at 1 km/h, incremented by 1 km/h every 30 seconds until the speed reached 5 km/h, and stayed at 5 km/h for another 30 seconds. When

the data collection finished, the treadmill slowed down to 1 km/h and stopped in 5 seconds. Visual instructions were displayed on the WISE to give prompts to participants.

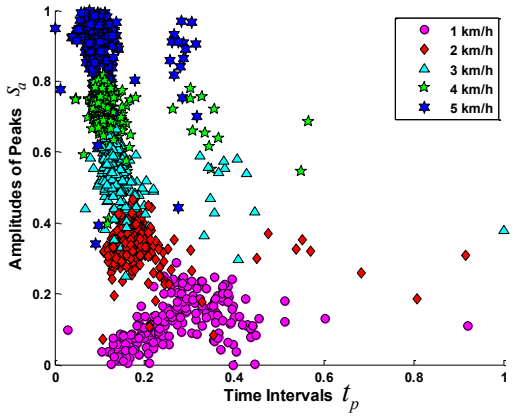


Figure 3-4: Features of the Training Set

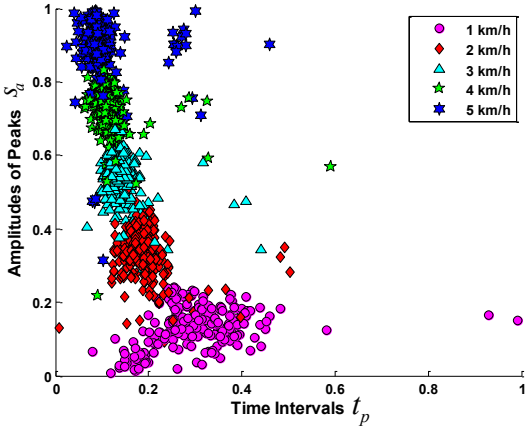


Figure 3-5: Features of the Testing Set

Table 3-1: Statistics of the Classification Performance

Feature	Multi-Class Precision	Multi-Class Recall	Average Accuracy
s_a	93.4 %	94.2 %	97.6 %
$s_a + t_p$	94.1 %	94.6 %	97.8 %

Table 3-2: The Mean Value and the Standard Deviation of the Error between the Average Foot Position and the Captured Trunk Position

Participant	1	2	3	4	5	6	7	8
Mean (m)	0.03	0.001	0.007	0.004	-0.004	-0.01	-0.007	-0.008
Std (m)	0.033	0.03	0.028	0.025	0.021	0.028	0.032	0.046

The data of three participants were not recorded properly due to the IR marker failure (subsequently corrected for Experiment 3-2 by using improved Velcro fasteners). Thus, I used the data of the first sessions of the rest of the 8 participants (3 males, 5 females, age: 21 - 26) as the training set and the data of the second sessions of the 8 participants as the testing set. To test the efficiency of the proposed algorithm, the step segmentation and feature extraction algorithms were applied on both sets. The extracted features vectors for the training set and the testing set are plotted in Figure 3-4 and Figure 3-5, respectively. The distinct clustering of feature vectors indicates that it is possible to classify them using the KNN algorithm. During testing, a KD-Tree was built by using the extracted feature vectors from the training set. The extracted feature vectors were classified using the KD-Tree to obtain the intended walking speed estimates $\hat{s}_w$.

The ground truth label of a feature vector was the true speed of the treadmill at which the foot positions were captured. I compared the result of the classification with its ground truth label to determine whether the result was correct or not.

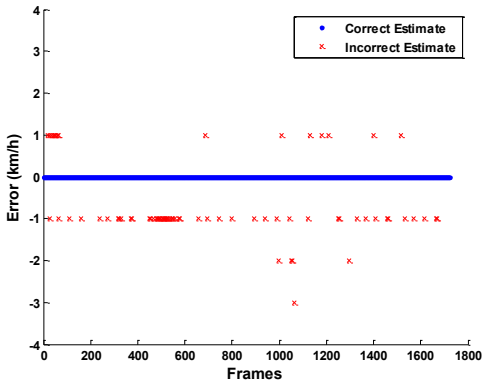


Figure 3-6: Error Pattern of Misclassification

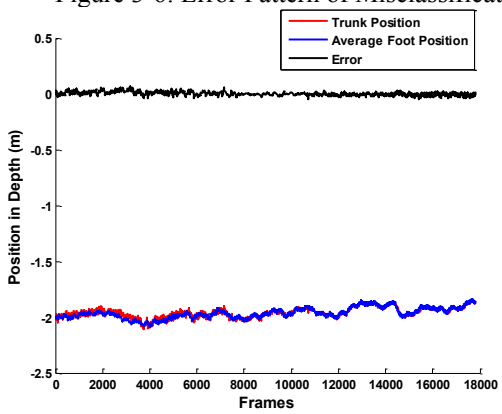


Figure 3-7: Trunk Position *vs* Average Foot Position

3.3.4 Results

I evaluated the KNN-based algorithm in terms of the multi-class recall ($Recall_M$), the multi-class precision ($Precision_M$) and the Average Accuracy for all 5 classes (Sokolova and Lapalme, 2009) (as shown in Table 3-1):

$$Recall_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l} \tag{3.4}$$

$$Precision_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l} \tag{3.5}$$

$$Average\ Accuracy = \frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}}{l} \tag{3.6}$$

where tp_i, tn_i, fp_i and fp_i denote true positive, true negative, false positive and false negative of class i and l is the total number of classes.

The evaluations were performed using the peaks of foot velocity s_a alone and using both the peaks of foot velocity s_a and the time interval t_p. For all measures, using s_a and t_p both generated better results than using s_a alone. The result demonstrates that using both features increased the classification performance by 0.2% compared to using foot velocity s_a alone.

To visualize the error pattern of misclassification, I plotted the estimated speed values for all participants, as shown in Figure 3-6. From it, I observed that the proposed algorithm tended to underestimate the actual walking speed, since the number of incorrect estimates at the error level of -1 km/h was obviously larger than that of 1 km/h. In the worst case (only a single frame), the classification produced an estimate 3 km/h slower than the actual walking speed.

To evaluate whether the average foot position is a reliable estimate of the trunk position, I calculated the mean value and the standard deviation of the error between the average foot position and the captured trunk position from the testing set (Table 3-2). In the worst case as shown by participant P1, the mean error was 0.03 m and standard deviation was 0.033 m. Figure 3-7 also presents exemplar curves and shows that the average foot position closely followed the trunk position and the error between these two curves

was minimal. The result demonstrates that the average foot position can be used to estimate the trunk position.

3.4 Experiment 3-2: Evaluation of the Usability of the Locomotion Interface

3.4.1 Introduction

The purpose of Experiment 3-2 was to evaluate the usability of the locomotion interface through a user study using the WISE (Figure 3-8). The participants' task was to pursue a rolling ball in the virtual environment by walking on the treadmill and try to maintain the

Figure 3-8: The WISE Running Experiment 3-2

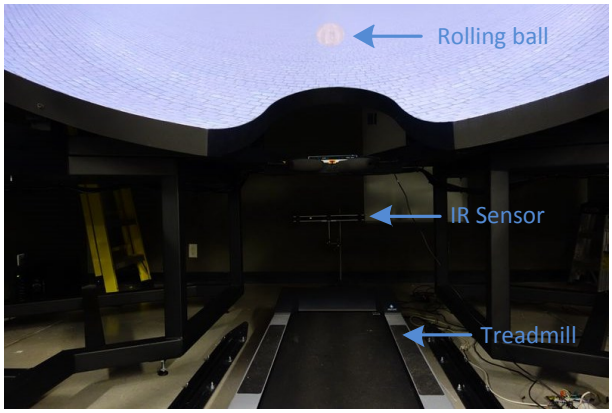


Figure 3-9: The Setup for Experiment 3-2

distance between the viewpoint and the ball to the initial distance of 5 m. The metric for usability was defined as the mean value and the standard deviation of the distance between the virtual viewpoint and the rolling ball, subtracted by the initial distance 5 m. Thus, the closer these two values to zero, the more usable the locomotion interface it is.

3.4.2 Participants

Ten people (8 male, 2 females, age: 18 - 28) participated in the study and none had participated in Experiment 3-1. Informed consent was obtained from all participants in accordance with a protocol approved by the Human Participants Review Subcommittee at York University.

3.4.3 Procedure

In Experiment 3-2, the IR sensor was also placed approximately 1 m in front of the treadmill (Figure 3-9) and participants were asked to attach two IR markers on the toe caps of their shoes. I designed a virtual scene in which the participants were asked to pursue a rolling ball using the locomotion interface. The speed of the ball was unpredictably varied between 1 km/h, 2 km/h and 3 km/h with changes in speed occurring at predefined key frames. The acceleration of the ball was set to 0.176 m/s^2, which matched the acceleration of the treadmill. The virtual scene was rendered on the WISE in monocular mode with a frame rate of 60 Hz. A KD-Tree was built by using the extracted feature vectors from the training set in Experiment 3-1 for speed estimation. The positional gain k_p in equation (3.3) was set to 3 and the reference position p_{ref} was set to 1.8 m empirically. For safety

reasons, I limited the range of speed command to 0.2 km/h minimum and 3.5 km/h maximum. The participants' task was to try to maintain the distance between the virtual viewpoint and the ball to the initial distance of 5 m. To provide a visual cue for participants to judge the speed and the distance of the ball, the color of the ball was updated on a per-frame basis such that the color would gradually become red if a participant was too far away from it or blue if too close using the following equations:

$$d = (p_b - p_v) / 5 - 1 \tag{3.7}$$

where p_b is the position of the ball, p_b is the position of the viewpoint, d is the distance between the ball and the viewpoint normalized by the 5 m target distance between the ball and the viewpoint.

$$(R, G\ B) = \begin{cases} (1, 1-d, 1-d) & if\ \ d \geq 0 \\ (1+d, 1+d, 1) & otherwise \end{cases} \tag{3.8}$$

where (R, G, B) are the color channels and $0 \leq R \leq 1.0, 0 \leq G \leq 1.0, 0 \leq B \leq 1.0$.

Since the length of the treadmill belt was limited and it was impractical to increase walking speed by increasing step length, the participants were instructed to increase their step frequency and foot swing speed if they intend to accelerate and conversely decrease their step frequency and foot swing speed for deceleration. Each participant was asked to perform 4 sessions of walking using the same scene and each session lasted 230 seconds. To ensure the experiment was conducted in the same condition for all participants and all sessions, the start and the stop of the treadmill and the ball were synchronized and controlled by the experiment application software. When the researcher started the experiment, a countdown timer was shown on the display and the data collection began at the same

time. The treadmill and the ball started to move simultaneously after a 10-second count. 3 seconds prior to the end of the experiment, another countdown timer was shown on the display and counted 3 seconds before the experiment stopped. The data collection was immediately stopped at 230 seconds and the ball and the treadmill were stopped subsequently. Given such experimental conditions, theoretically, a participant should be able to control the viewpoint through the locomotion interface such that the curve of the position of the viewpoint perfectly matched the trajectory of the rolling ball while maintaining the 5 m reference distance.

3.4.4 Results

Figure 3-10 shows an example series of the trajectories of the ball and the virtual viewpoint. The 5 m distance offset was subtracted from the trajectory of the ball point-wise to show that the participant was able to maintain the 5 m distance and matched the trajectory of the viewpoint to that of the rolling ball. Figure 3-11 shows an example of the speed changes of the ball and the viewpoint and the speed commands issued by the controller in a single experiment session by a participant. I observed that in some cases the speed of the viewpoint did not reach the value given by a speed command due to the modest maximum acceleration of the treadmill. The performance of the participants and the locomotion interface was evaluated in terms of the error of the distance between the virtual viewpoint

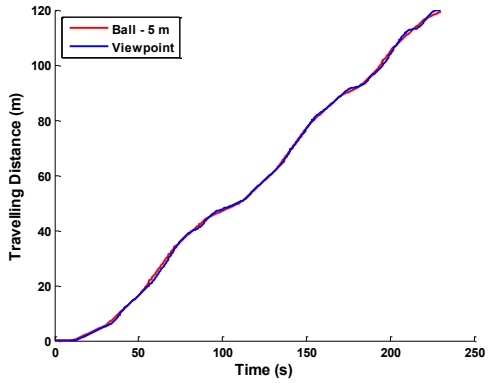


Figure 3-10: Exemplar Trajectories of the Ball and the Viewpoint

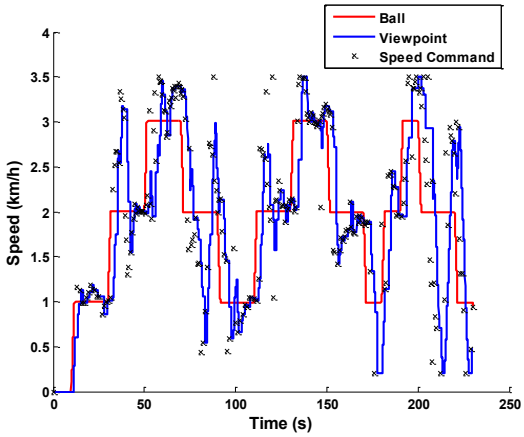


Figure 3-11: Experimental Data Traces

and the ball with respect to the reference distance 5 m. This point-wise error was calculated by the following equation:

$$\varepsilon = p_b - p_v - 5 \tag{3.9}$$

where p_b denotes the position of the ball and p_v denotes the position of the viewpoint. I then calculated the mean value and standard deviation of the point-wise error from the 4th session of all participants. The results for all participants are shown in Figure 3-12 using an error plot, which shows that participants P1, P3, P5, P6 and P8 performed relatively

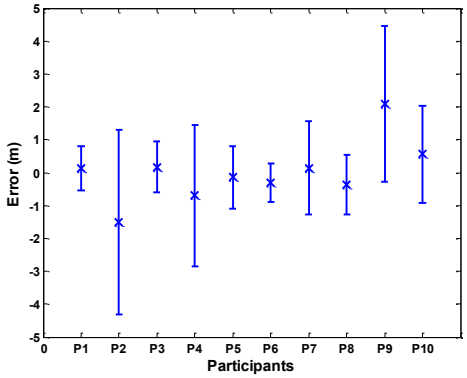


Figure 3-12: Performance of the Participants

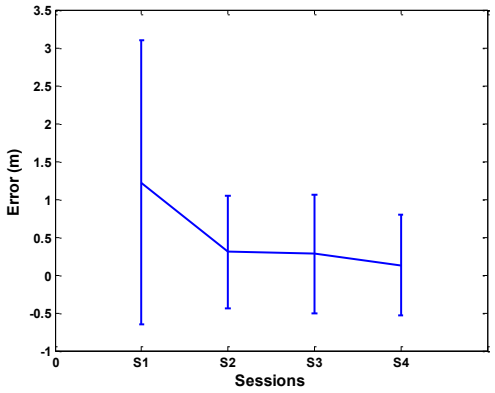


Figure 3-13: The Learning Effect of a Participant

better than the other 5 participants. The mean value of the error shows on average whether a participant is leading ahead or falling behind of the 5 m distance to the ball and the standard deviation reflects the interval in which the participant oscillates while maintaining the 5 m distance or the errors or delays at speed transition.

Several factors may lead to the error. These include a participant's attention, the ability to adapt to self-paced treadmill walking and control of gait, the accuracy of the speed estimation algorithm and the hardware limitations of the treadmill. A participant's

attention on the ball is critical for the experiment. If attention is not focused on the ball and the task, then the distance and the speed of the ball cannot be judged and controlled. In addition, walking on a self-paced treadmill is different than walking over-ground. Thus, participants must quickly adapt themselves to the walking condition and learn to control the walking speed through the changing of foot swing speed and step frequency instead of step length. The accuracy and the resolution of the speed estimation algorithm may be improved if I only use the data collected from a specific participant to classify that participant's data during the user study. A major hardware limitation of the treadmill is its low acceleration, which means it takes time for the treadmill reach the speed command sent by the host computer. Several participants reported the lack of responsiveness due to the issue.

The results of two participants had clear learning effects as the mean errors across four experimental sessions were monotonically decreasing. Figure 3-13 shows an example of a participant that had such a clear learning effect. Five participants had smaller mean errors in the fourth sessions compared to the first sessions, but the mean errors fluctuated across sessions. The mean errors of three participants in the fourth sessions were larger than that of first sessions. I suspected that these participants needed to have more training and practice sessions to improve their performance and minimize the mean errors.

3.5 Discussion

In this chapter, I presented a machine learning approach for implementing a locomotion technique based on a conventional 1-D treadmill. I used the locomotion interface for interacting with the WISE and conducted experiments evaluating its usability through a novel

user study. For future research, I will implement a turning strategy (Vijayakar and Hollerbach, 2002) for the locomotion interface. The improved locomotion interface can be used for the study of target interception in VR environments. Meanwhile, the speed estimation algorithm may also be used with other types of sensors, such as inertial measurement units, for estimating walking speed either over-ground or on treadmill. Finally, an empirical comparison of the proposed approach with other classic approaches of locomotion based on 1-D treadmills also is needed.

The proposed locomotion technique may have many different utilities. For example, it can be used as an educational platform for people to experience walking in VR. More unusual scenes can be added to the VR system, including famous tourist spots, cartoon worlds or surfaces of other planet, for people to have an immersive experience to walk in the places that they may not have access to. Finally, such a locomotion technique may serve as a rehabilitation platform for patients to practice walking as locomotion in VR is more interesting than walking on a treadmill alone and the walking performance and the improvement of patients can be monitored by VR tracking devices.

Chapter 4
The Role of Stereopsis in Avoiding Virtual Obstacles While Walking

4.1 Introduction

Walking is a daily routine for us. We usually encounter and effortlessly avoid obstacles and gaps during walking. To avoid being tripped, we adjust our footsteps to step over hazards based on the visual information provided by our eyes. Stereopsis is important for visually guided behavior and has been shown to aid hand-eye coordination (Fielder and Moseley, 1996). Many studies have been conducted to study the relationship between stereopsis and the performance of tasks related to upper limbs. Stereopsis was originally thought to be not very helpful for locomotion, such as walking and running, as a period of steady viewing is required to reach a certain level of precision (McKee *et al*., 1990). However, studies have shown that stereoscopic viewing provides advantages over non-stereoscopic viewing in terms of more accurate lower limb movements (Patla *et al*., 2002; Loomis *et al*., 2006; Hayhoe *et al*., 2009; Chapman *et al*., 2012), as reviewed in Section 2.2.2.

While these studies have been conducted to investigate the role of stereopsis in locomotion, a primary limitation was that the scene setups were relatively simple, and the experiments were usually conducted in an indoor setup with limited walking space. In everyday life, we often walk continuously over longer distances. Experiments conducted with limited walking distance may not reflect what we experience in real-life. Thus, the influence of the stereopsis on continuous walking is still uncertain.

Virtual Reality (VR) systems combined with mechanical repositioning techniques (Nilsson *et al*., 2013) (to reposition a user to the center of the tracked physical space using mechanical devices, such as treadmills (Iwata, 1999; Souman *et al*., 2008), foot platforms (Iwata *et al*., 2001), pedaling devices (Allison *et al*., 2000) and spheres (Medina *et al*., 2008), *etc*.) provide us with a unique opportunity to simulate large open environments. These systems enable people to walk over long distances with their motion recorded in a limited physical space. Thus, these are promising platforms to investigate the influence of stereopsis on gait parameters in continuous walking. In this study, I presented two experiments in VR to examine the effects of stepping over obstacles and gaps in continuous walking under stereoscopic and non-stereoscopic viewing conditions. These experiments were conducted using a novel immersive projected display, known as the Wide-Field Immersive Stereoscopic Environment (WISE). This display together with a 1-D treadmill allowed us to implement straight line walking in VR. Virtual scenes that presented obstacles and gaps for the experiments based on the setup were designed.

4.2 Method

4.2.1 Hardware and Software of the VR system

The virtual environments for the experiments were presented on the large-scale curved projected display – the WISE. The images rendered on the display were cast and seamlessly merged by eight stereoscopic overlapping projectors, with geometry correction (warping), blending and luminance calibration performed in hardware. Each projector was driven by a client machine (HP Z820 Workstation with Nvidia Quadro k5000 graphics card) in a real-

time rendering cluster. The rendering and the synchronization between the host machine (HP Z820 Workstation with Nvidia Quadro k5000 graphics card) and the client machines are handled by the VR software Worldviz Vizard 5.7. Stereoscopic viewing was presented through the Christie shutter glasses at a refresh rate of 60 Hz for each eye. The Worldviz PPT Eyes tracker was mounted on the top of the glasses frame to track head moevments. Body movements of participants, including head motion and foot motion, were captured using the Worldviz PPT system, which used infrared (IR) cameras to capture IR light emitted by markers attached on body parts to be tracked. Three IR cameras were mounted on the top of the display facing the ground to track head positions while another three IR cameras were mounted under the display facing the treadmill to track foot positions. The 3-D positions of the tracked IR markers were calculated by the Worldviz PPT Studio and was shared with the Worldviz Vizard simulation through the Virtual-Reality Peripheral Network (VRPN) (Taylor *et al*., 2001). The PPT Eyes were equipped with two IR markers mounted on the top of the glasses frame. This enabled the 3-D position and orientation of the PPT Eyes (i.e. the position and the orientation of the head) to be tracked. Disparity between eyes and perspective transformation of a scene were generated based on tracked head position in real-time. A commercial 1-D treadmill, LifeSpan TR5000-DT5, was used as the walking platform. The top surface of the treadmill belt had a size of 51 cm (W) × 142 cm (L). The treadmill was controlled through Universal Asynchronous Receiver/Transmitter (UART) controllers by a host machine using a baudrate of 4800 bit/s. The speed of the treadmill was queried periodically at a frequency of 5 Hz. The queried

value was synchronized to the speed of the virtual viewpoint to create egocentric motion in the virtual environment. The position of the virtual viewpoint and the positions of tracked body parts were recorded at a frequency of 60 Hz. The experiment software application that integrated the presentation of virtual environments, hardware control and data recording was implemented using Python 2.7. This integrated VR system allows participants to perform linear walking with their movements recorded.

4.2.2 Feature Extraction for Gait Analysis

Recorded foot positions from the IR markers mounted on participants' ankles were used for data analysis. I estimated gait parameters extracted from tracked foot positions to study how stereopsis affects people's walking performance. I explain my method for extracting gait parameters of walking on a treadmill in this section.

The position of a tracked single foot is denoted as a 3-D vector $p_f = (x_f, y_f, z_f)$, where x_f, y_f and z_f are lateral, vertical and depth positions, respectively. A sequence of recorded foot positions is represented as $P_f = p_{f1}p_{f2} \dots p_{fi}$, where i is the index for a 3-D vector p_f. Contrary to over-ground walking in which the z-position of a person's foot monotonically increases or decreases, when a person walks on a treadmill their feet perform reciprocating motion in terms of depth and the tracked z-position oscillates as opposed to over-ground walking. As I aim to analyze participants' gait in virtual environments, it is necessary to match the tracked physical foot position to the equivalent virtual foot position in the virtual environment. This can be done by performing a transformation on the tracked

physical foot position with respect to the position of the virtual viewpoint, which is represented as p_v . A sequence of positions of virtual viewpoint is represented as $P_v = p_{v1}p_{v2} \dots p_{vi}$. Recall that I synchronized the speed of the virtual viewpoint with respect to the speed of the treadmill. Assume that a person walks on the treadmill with their head position maintained at the center of the treadmill in tracked physical space, their head position in the virtual environment is essentially the position of the virtual viewpoint p_v as the person walks forward. As the changes of foot positions are relative to the head position in tracked physical space, the transformation between foot positions in tracked physical space and in the virtual environment can be performed by adding the sequence of the foot positions P_f and the sequence of the positions of virtual viewpoint P_v. The *x*-component and *y*-component in p_v were set to zero for transformation, with $p_{vi} = (0, 0, z_{vi})$, since it was not necessary to transform the *x*-component and *y*-component of P_f and I only needed to recover the depth of P_f. This gives:

$$P_t = P_f + P_v \tag{4.1}$$

where P_t is the transformed foot position sequence with $P_t = p_{t1}p_{t2} \dots p_{ti}$ and $p_{ti} = (x_{ti}, y_{ti}, z_{ti})$. After the transformation, instantaneous foot velocity V_t was calculated from P_t using 2-point difference. P_t and V_t were smoothed using 2nd order Butterworth filters with a cut-off frequency of 1 Hz to remove noise. The transformation and filtering were performed on recorded position data of both feet.

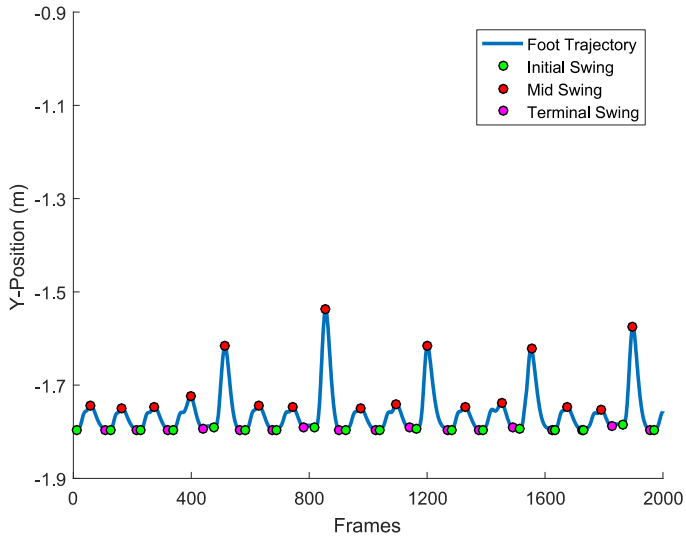


Figure 4-1: Exemplar Segmentation of Gait Cycles

A gait cycle is defined as two consecutive heel strikes of the same foot (Gamble and Rose, 1994). To extract gait cycles for analysis, I examined the sequence of transformed vertical foot positions y_{ti}. This is similar to the approach presented in Chapter 3 that used foot speed in depth to segment steps. Specifically, I located local minimums between peaks to segment a gait cycle by first applying a high threshold τ_h to the data sequence of y_{ti} (the median of the data sequence of y_{ti} was set as τ_h). From the thresholded data points, the algorithm used gradient descent to locate the initial swing s_{init} and the terminal swing s_{term} (which correspond to frames). The gradient descent stopped whenever it reached the minimum threshold τ_l or it found that the gradient was ascending, indicating a different gait cycle was detected. Once the frames corresponding to initial swing s_{init} and the terminal swing s_{term} were successfully located, the maximum position value in the interval between these two swing points was selected and its index was considered

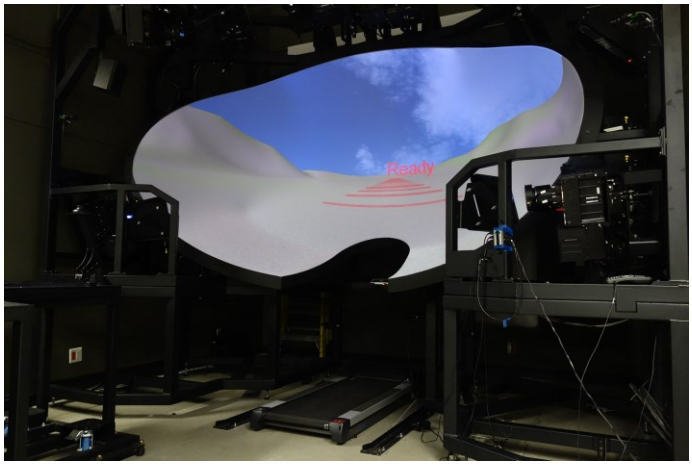


Figure 4-2: The Setup of Experiment 4-1

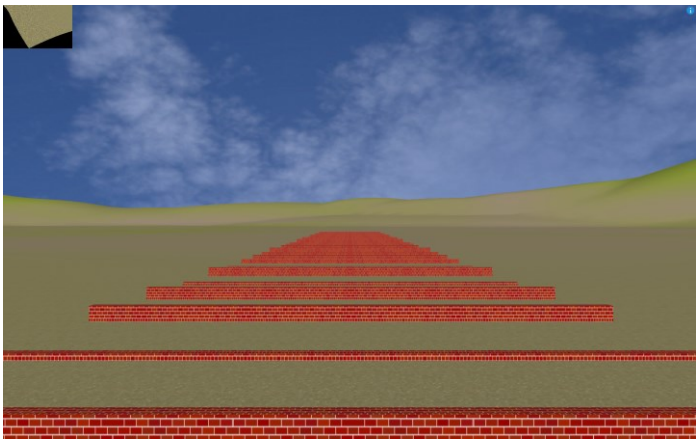


Figure 4-3: Console View of Experiment 4-1

as the mid swing s_{mid}. Figure 4-1 shows an example of segmented gait cycles on a single foot trajectory that stepped over obstacles. This approach was in general more robust than applying a single threshold to segment gait cycles. With the detected initial swing s_{init}, mid swing s_{mid} and terminal swing s_{term}, it was easy to calculate gait parameters, such as stride length and stride height. I then merged the gait cycles segmented from the position data of both feet based on the sorted z-position of the mid swing s_{mid} of the gait cycles to

obtain a single sequence of gait cycles in an ascending order of z-positions. Distinguishing foot position data between left foot and right foot was not necessary for further data analysis.

I used the minimum distance classifier (Lin and Venetsanopoulos, 1993) to register the merged gait cycles with respect to obstacles or gaps presented in experiments. In other words, the responses (gait cycles) were associated with stimuli (obstacles with different heights or gaps with different depths) through the classification. This was done by calculating the Euclidean distances between the z-position of the mid swing s_{mid} of the merged gait cycles G_j and the centers of the z-position of obstacles or gaps C_i:

$$R_i = \underset{i}{\operatorname{argmin}} \left(\left\| G_j - C_i \right\|_2 \right) \tag{4.2}$$

where j is the index of the z-position of the mid swing s_{mid} of the merged gait cycles G_j and i is the index of the centers of the z-position of obstacles or gaps C_i, respectively. R_i is the resulting index of an obstacle or gap to be associated with its corresponding gait cycles G_j. In practice, this equation was solved by looping through all combinations of z-positions of gait cycles and z-positions of obstacles or gaps. The pairs with the minimum Euclidean distance were registered together. When the registration was completed, I was able to evaluate a specific gait cycle with respect to the obstacle or the gap that it covered. I defined the following metrics to evaluate the gait performance of participants:

- **Stride length l_s:**

  The z-distance between initial swing, s_{init}, and terminal swing, s_{term}.

- **Stride height h_s:**

The difference in height of the foot at mid swing, s_{mid}, and when the foot was planted (the average of y-positions of initial swing, s_{init}, and terminal swing, s_{term}).

- **Foot lifting distances to obstacles or gaps d_l:**

The difference in z-distance of the foot at initial swing, s_{init}, and the front face of an obstacle or the front edge of a gap.

- **Foot planting distances to obstacles or gaps d_p:**

The z-distance between the foot at terminal swing, s_{term}, and the back face of an obstacle or the back edge of a gap.

- **Foot clearance to obstacles d_c:**

The y-distance of the foot at mid swing, s_{mid}, to the top of an obstacle. Foot clearance to gaps were not assessed as this is the same parameters as stride height h_s, with an added deepness of gaps fixed as 0.5 m.

- **Foot velocity of mid-swing s_f:**

The instantaneous speed of the foot at mid swing, s_{mid}, obtained by calculating the Euclidean norm of the y-component and the z-component of foot velocity V_t.

- **Number of strides n_s:**

The number of strides that were taken during a single walking trial.

- **Number of collisions n_c:**

The number of collisions happened between the transformed foot position P_t and the bounding boxes of obstacles or gaps during a single walking trial. As people were unable to step into a gap physically, the bounding boxes of a gap was modeled with a low height of 0.01 m above the ground surface to determine the occurrence of collisions.

4.3 Experiment 4-1: Stepping Over Obstacles

4.3.1 Introduction

The goal of experiment 4-1 was to investigate whether stereopsis provides advantages when people step over obstacles in VR environments.

4.3.2 Design

In this experiment, I designed an outdoor environment that had a valley and a skydome, using Autodesk 3ds Max 2016, shown in Figure 4-2. A console view of the scene on the host machine is shown in Figure 4-3. The texture for the valley was manually blended from a grass texture and a gravel texture while the texture for the skydome was a high definition picture that captured a bright sky with few white clouds. The obstacles were brick-textured cubic objects. The width (x-axis) and depth (z-axis) of the obstacles were fixed as 10 m and 0.2 m, respectively. The heights (y-axis) of these obstacles had three different values, which were 0.1 m, 0.2 m and 0.3 m. These constituted three different conditions for the experiment. Each condition was repeated ten times for each scene. Thus, in total, there were thirty objects in an experimental scene, with the order of the objects randomized. The distance between the participant to the front face of the first obstacle was 5 m. The distance between

the back face of an obstacle and the front face of its immediate successive obstacle was 3 m. This gave participants an adequate amount of distance to walk normally and adjust their footsteps before stepping over the next obstacle. The total length of each walking path was approximately 100 m. Participants were expected to perform constant speed linear walking in the virtual environment.

Each walk through an experimental scene with a random order of generated obstacles was considered as a single trial. Participants were first asked to perform two trials under the stereoscopic viewing condition as practice to get familiar with the hardware and the virtual environment. Then, participants were asked to perform two trials under the stereoscopic viewing condition and two trials under the non-stereoscopic viewing condition. The order of trials under the stereoscopic viewing condition and the non-stereoscopic viewing condition were counter-balanced to control for order effects. Five participants followed an order of viewing conditions of SSNN, where S denotes the stereoscopic viewing condition and N denotes the non-stereoscopic viewing condition while another five participants followed an order of NNSS. For the non-stereoscopic viewing condition, participants were also asked to wear the PPT Eyes, but modeled distance between two eyes was set to zero.

4.3.3 Participants

Ten people (7 males, 3 females, age: 24 - 39, height: 1.59 - 1.90 m) participated in the experiment. All had normal or corrected-to-normal vision. Stereo acuity of participants was verified using the Randot Stereotest (Stereo Optical Company, Inc. Chicago IL). All had

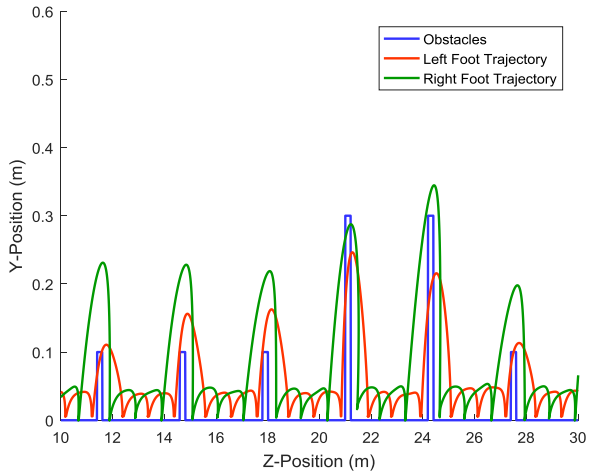


Figure 4-4: Foot Trajectories on Stepping over Obstacles

good stereo acuity ($\leqslant$ 50 seconds of arc). Informed consent was obtained from all participants in accordance with a protocol approved by the Human Participants Review Subcommittee at York University.

4.3.4 Procedure

During experimental sessions, participants wore the PPT Eyes on their head and two IR markers, one on each ankle, and stood on the treadmill. For a single experimental trial, when the experiment was started, a ten-second countdown timer was shown on the WISE and the data collection started at the same time. The belt of the treadmill automatically began to move when the timer counted to zero. Then, the treadmill accelerated to 2 km/h and maintained this speed through the experimental trial. Participants were asked to accommodate their walking speed to the speed of the treadmill and step over obstacles when they felt necessary. When the virtual viewpoint passed the last obstacle in the virtual scene,

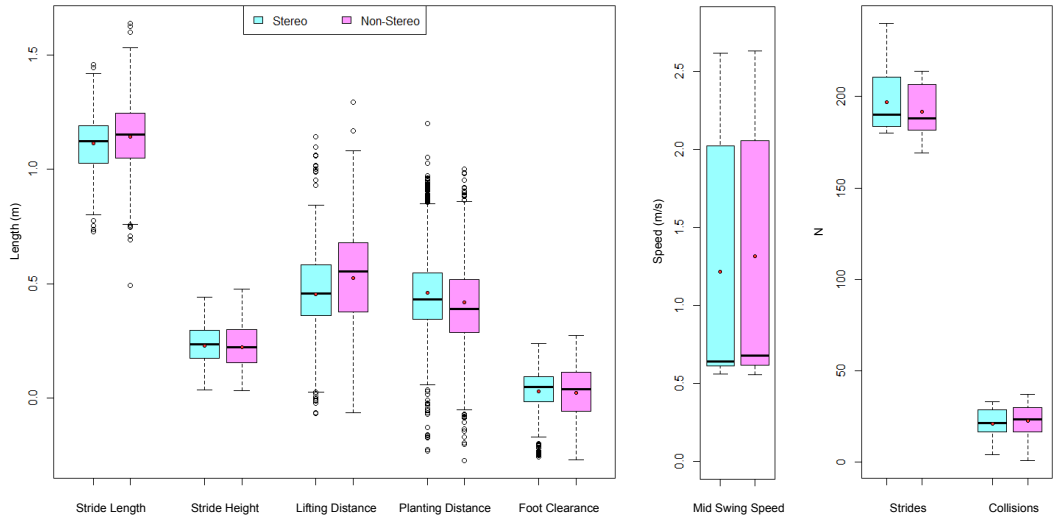


Figure 4-5: Gait Parameters on Stepping over Obstacles by Viewing Condition (red dots denote mean values; the boxes of the number of strides and the number of collisions denote the data distribution of that of all walking trials of each viewing condition across participants; for other gait parameters, the boxes denote the data distribution from the gait parameters of all gait cycles that covered an obstacle for each viewing condition)

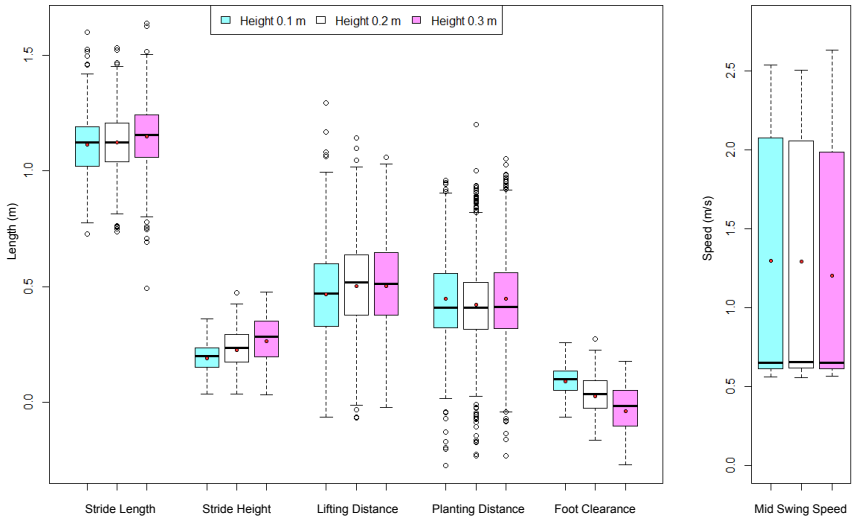


Figure 4-6: Gait Parameters on Stepping over Obstacles by Height Level (red dots denote mean values; the boxes of these gait parameters denote the data distribution from the gait parameters of all gait cycles that covered an obstacle for each level of obstacle height)

Table 4-1: Results of the Linear Mixed-Effects Models Analyses on Stepping over Obstacles (significant p-values are in bold and shaded)

		l_s	h_s	d_l	d_p	d_c	s_f
Viewing Condition	$F(1,1185)$	18.99	0.98	38.19	14.00	0.98	5.66
	p	**<0.001**	0.322	**<0.001**	**<0.001**	0.322	**0.017**
	η_p^2	0.016	0.001	0.031	0.012	0.001	0.005
Height Level	$F(2,1185)$	9.19	111.58	3.73	2.39	325.68	2.29
	p	**<0.001**	**<0.001**	**0.024**	0.092	**<0.001**	0.102
	η_p^2	0.015	0.158	0.006	0.004	0.355	0.004
Viewing Condition × Height Level	$F(2,1185)$	0.04	5.81	1.35	1.25	5.81	0.91
	p	0.964	**0.003**	0.258	0.288	**0.003**	0.401
	η_p^2	0.000	0.010	0.002	0.002	0.010	0.002

		n_s	n_c
Viewing Condition	$F(1,29)$	5.99	0.81
	p	**0.021**	0.376
	η_p^2	0.171	0.027

another three-second countdown timer was shown on the WISE, informing participants that the experiment would finish soon. The experiment ended when the timer counted to zero, with the data collection and treadmill stopped simultaneously.

4.3.5 Results and Discussion

A segment of recorded foot trajectories when stepping over obstacles can be seen in Figure 4-4 for illustration. To analyze the experimental data, I applied the method described in Section 4.2.2 on recorded foot positions to extract gait parameters using Matlab 2016a. I then performed statistical analysis on the extracted gait parameters defined in Section 4.2.2

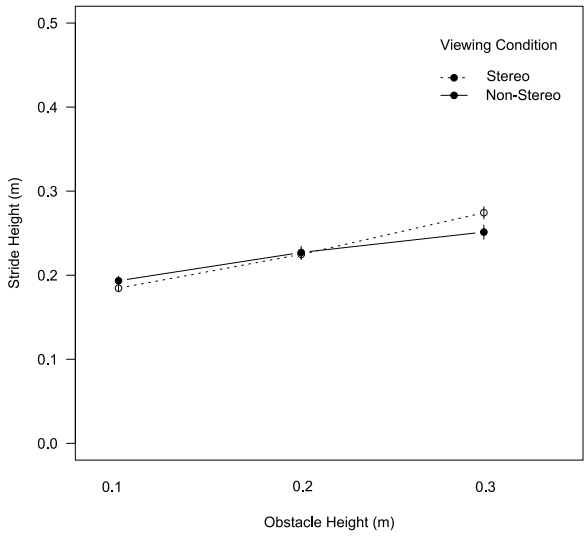


Figure 4-7: Interaction Effect on Stride Height (error bars denote the standard error of the mean)

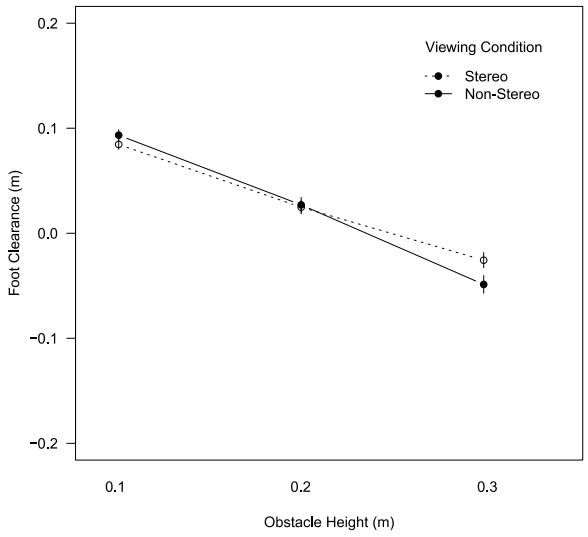


Figure 4-8: Interaction Effect on Foot Clearance (error bars denote the standard error of the mean)

using R 3.4.2. The Linear Mixed-Effects Models analyses (Package NLME in R) were used to study the effects of the experiment. Effect sizes were reported using partial eta squared

η_p^2 (estimated from repeated-measures ANOVA analyses of the same form as the Linear Mixed-Effects Models analyses). The independent factors involved were viewing conditions (stereoscopic and non-stereoscopic) and height levels (0.1 m, 0.2 m and 0.3 m) of the obstacles while the dependent factors were the gait parameters. Post-hoc pairwise comparisons were performed using Tukey's range tests. Figure 4-5 and Figure 4-6 show box plots on gait parameters and Table 4-1 summarizes the results of the Linear Mixed-Effects Models analyses.

I found significant effects on stride length l_s $(F(1,1185) = 18.99, p < 0.001, \eta_p^2 = 0.016)$, foot lifting distance to obstacles d_l $(F(1,1185) = 38.19, p < 0.001, \eta_p^2 = 0.031)$, foot planting distance to obstacles d_p $(F(1,1185) = 14.00, p < 0.001, \eta_p^2 = 0.012)$ and mid swing speed s_f $(F(1,1185) = 5.66, p = 0.017, \eta_p^2 = 0.005)$. Under the stereoscopic viewing condition, stride length was smaller than for the non-stereoscopic viewing condition. Stride length under stereoscopic viewing was more accurate as a smaller stride was sufficient to cover an obstacle. In Figure 4-5, I found that the mean value of the foot lifting distance to obstacles was smaller under stereoscopic viewing condition than non-stereoscopic viewing condition. I also found that the mean value of the foot planting distance to obstacles was larger under stereoscopic viewing condition than non-stereoscopic viewing condition. The result showed stereoscopic viewing was beneficial as, in reality, if we wish to safely step over an obstacle, we would step as closely to the front side of the obstacle as possible with one foot and walk over it with the other foot to plant far from the back side of the obstacle. It was obvious that stereopsis

helped to realize this aim during walking. The mean value of mid swing speed was lower under stereoscopic condition than non-stereoscopic condition.

Although there was no significant effect alone on stride height h_s between viewing conditions, there was a significant interaction effect between viewing conditions and height levels on stride height h_s ($F(2, 1185) = 5.81, p = 0.003, \eta_p^2 = 0.010$) (Figure 4-7), which was consistent with the significant interaction effect between these factors on foot clearance to obstacles d_c ($F(2, 1185) = 5.81, p = 0.003, \eta_p^2 = 0.010$) (Figure 4-8). But the interaction effects on both parameters were weak. For both stride height h_s and foot clearance to obstacles d_c, Tukey's range tests showed that there was a difference between stereoscopic viewing *vs* non-stereoscopic viewing on height level 0.3 m for both stride height h_s and foot clearance to obstacles d_c but not for 0.1 m and 0.2 m obstacle height. The mean value of the stride height h_s for the 0.3 m obstacle under stereoscopic viewing was 0.27 m while the mean value under non-stereoscopic viewing was 0.25 m, which showed that people tended to lift their feet higher under stereoscopic viewing condition when they encountered obstacles with a height of 0.3 m. This may imply that fewer tripping hazards would happen when walking with stereoscopic vision as their feet were lifted higher. Similarly, I also found that the mean value of foot clearance to obstacles under stereoscopic viewing was higher than non-stereoscopic viewing. However, there were no interaction effects on other gait parameters. The mean value of stride height h_s under both stereoscopic viewing and the non-stereoscopic viewing was generally insufficient for step-

ping over obstacles. This may reflect that in virtual environments, there was no actual tripping consequence when the stride height was lower than the height of obstacles. Alternatively, when walking on a moving treadmill in a virtual environment, people may have acted more cautiously to maintain their balance. Thus, their feet were not lifted high enough for the obstacles with a height of 0.3 m.

A significant effect was found on number of strides n_s between viewing conditions ($F(1, 29) = 5.99, p = 0.021, \eta_p^2 = 0.171$). Walking under stereoscopic viewing resulted in more strides compared to non-stereoscopic viewing (Figure 4-5). Given that the total lengths of walking paths for all experimental trials were nearly the same, it suggested that the cadence under stereoscopic viewing was higher than non-stereoscopic viewing. This was also confirmed by shorter stride length under stereoscopic viewing compared to non-stereoscopic viewing. In addition, there was no significant effect on number of collisions n_c ($F(1, 29) = 0.81, p = 0.376, \eta_p^2 = 0.027$), which suggested that avoiding collision with obstacles was equally difficult between stereoscopic viewing and non-stereoscopic viewing in virtual environments.

For height levels, I found significant effects on stride length l_s ($F(2, 1185) = 9.19, p < 0.001, \eta_p^2 = 0.015$), stride height h_s ($F(2, 1185) = 111.58, p < 0.001, \eta_p^2 = 0.158$), foot lifting distance to obstacles d_l ($F(2, 1185) = 3.73, p = 0.024, \eta_p^2 = 0.006$), foot clearance to obstacles d_c ($F(2, 1185) = 325.68, p < 0.001, \eta_p^2 = 0.355$) but not on foot planting distances to obstacles d_p ($F(2, 1185) = 2.39, p = 0.092, \eta_p^2 = 0.004$) and mid swing speed s_f ($F(2, 1185) = 2.29, p = 0.102, \eta_p^2 = 0.004$). In Figure 4-6, I found

that for obstacles with a height of 0.3 m, participants' feet were not lifted high enough as the mean value of foot clearance was clearly negative. Tukey's range tests revealed that there were significant differences between three different height levels on stride height h_s and foot clearance to obstacles d_c; a significant difference between height level 0.1 m and height level 0.3 m on stride length l_s; and significant differences between height level 0.1

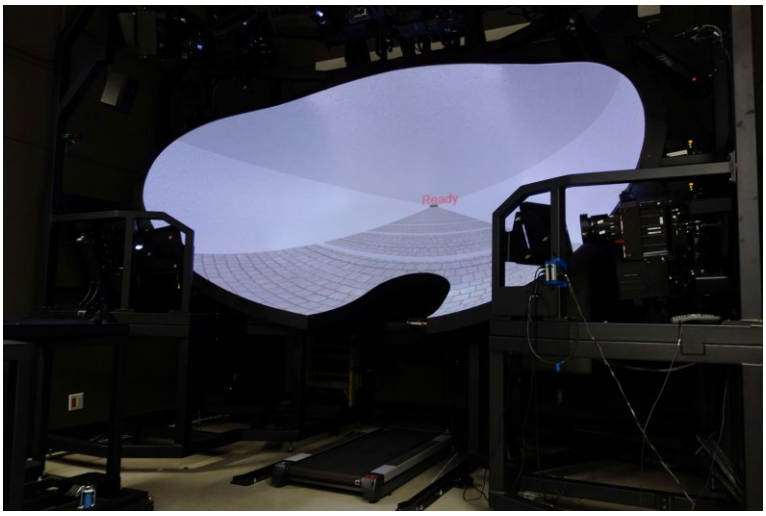


Figure 4-9: The Setup of Experiment 4-2

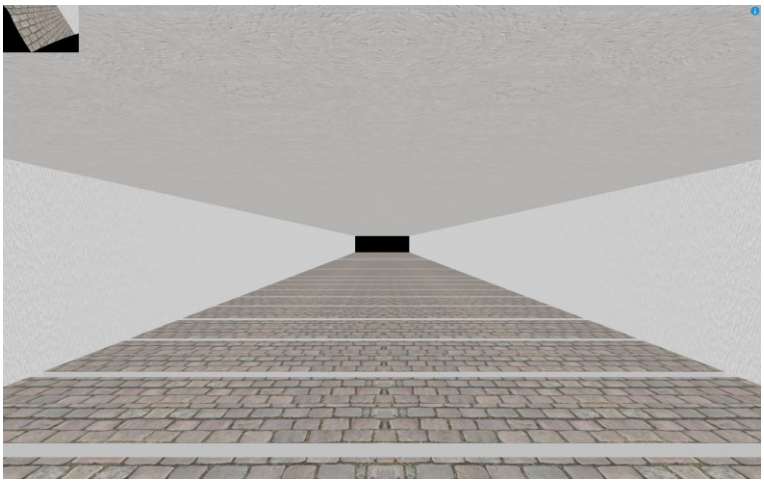


Figure 4-10: Console View of Experiment 4-2

m and height level 0.2 m and between height level 0.1 m and height level 0.3 m on foot lifting distance to obstacles d_l. Thus, people adjusted their footsteps when they encountered obstacles with different heights.

4.4 Experiment 4-2: Stepping Over Gaps

4.4.1 Introduction

The goal of experiment 4-2 was to investigate whether stereopsis provides advantages when people step over gaps in VR environments.

4.4.2 Design

In this experiment, I designed an indoor virtual environment that consisted of a ground surface with gaps, two side walls and a celling shown in Figure 4-9. A console view of the scene on the host machine is shown in Figure 4-10. These geometries were textured using different stone images to create contrasts between the ground, walls and the ceiling. Here, I referred the height of the ground surface to the bottom of the gaps as deepness (y-axis) and the distance between the front edge of a gap to the back edge of a gap as depth (z-axis). The width (x-axis) and the deepness (y-axis) of the gaps were fixed as 10 m and 0.5 m, respectively. The depth (z-axis) of the gaps had three different values, which were 0.2 m, 0.3 m and 0.4 m. As in the previous experiment, each condition (i.e. depth) was repeated ten times. Thus, in an experimental scene, there were thirty gaps in total and the order of the gaps was randomized. The distance between the participant and the front edge of the first gap was 5 m and the distance between the back edge of a gap and the front edge of its

immediate successor was 3 m. The total length of each walking path was approximately 100 m. Participants were also expected to perform constant speed linear walking in the virtual environment.

As in the previous experiment, each generated experimental scene was considered as a single trial and participants were asked to perform two training trials under stereoscopic viewing condition, subsequently followed by two experimental trials under stereoscopic viewing condition and two experimental trials under non-stereoscopic viewing condition, with the order of experimental trials counter-balanced as in the previous experiment.

4.4.3 Participants

Ten people (5 males, 5 females, age: 20 - 39, height: 1.58 - 1.79 m) participated in the experiment. All had normal or corrected-to-normal vision. Stereo acuity of participants was verified using the Randot Stereotest (Stereo Optical Company, Inc. Chicago IL). All had good stereo acuity ($\leqslant$ 50 seconds of arc). Informed consent was obtained from all participants in accordance with a protocol approved by the Human Participants Review Subcommittee at York University.

4.4.4 Procedure

Participants wore the PPT Eyes on their head and two IR markers, one on each ankle, and stood on the treadmill. For a single experimental trial, when the experiment was started, a ten-second countdown timer was shown on the WISE and the data collection started at the same time. The belt of the treadmill automatically began to move when the timer counted

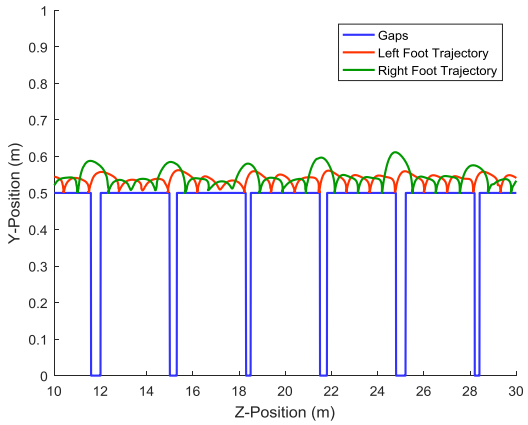


Figure 4-11: Foot Trajectories on Stepping over Gaps

to zero. Then, the treadmill accelerated to 2 km/h and maintained this speed through the experimental trial. Participants were asked to accommodate their walking speed to the speed of the treadmill and step over gaps when they felt necessary. When the virtual view-point passed the last gap in the virtual scene, another three-second countdown timer was shown on the WISE, informing participants that the experiment would finish soon. The experiment ended when the timer counted to zero, with the data collection and treadmill stopped simultaneously.

4.4.5 Results and Discussion

As in the previous experiment, I applied the method described in Section 4.2.2 on recorded foot positions to extract gait parameters. I then performed the Linear Mixed-Effects Models analyses (Package NLME) using R 3.4.2. Effect sizes were reported using partial eta squared η_p^2 (estimated from repeated-measures ANOVA analyses of the same form as the Linear Mixed-Effects Models analyses). The independent factors involved were viewing

conditions (stereoscopic and non-stereoscopic) and depth levels (0.2 m, 0.3 m and 0.4 m) of the gaps and the dependent factors were the gait parameters. Post-hoc pairwise comparisons were performed using Tukey's range tests. Figure 4-12 and Figure 4-13 show the box plots on gait parameters and Table 4-2 summarizes the results of the Linear Mixed-Effects Models analyses.

Table 4-2: Results of the Linear Mixed-Effects Models Analyses on Stepping over Gaps (significant p-values are in bold and shaded)

		l_s	h_s	d_l	d_p	s_f
Viewing Condition	$F(1,1185)$	13.01	4.87	1.94	8.68	5.35
	p	**<0.001**	**0.028**	0.164	**0.003**	**0.021**
	η_p^2	0.011	0.004	0.002	0.007	0.004
Depth Level	$F(2,1185)$	29.12	6.31	17.36	1.27	0.09
	p	**<0.001**	**0.002**	**<0.001**	0.281	0.913
	η_p^2	0.047	0.011	0.028	0.002	0.000
Viewing Condition × Depth Level	$F(2,1185)$	0.24	0.68	0.91	1.22	1.87
	p	0.789	0.508	0.404	0.297	0.154
	η_p^2	0.000	0.001	0.002	0.002	0.003

		n_s	n_c
Viewing Condition	$F(1,29)$	0.25	0.35
	p	0.618	0.561
	η_p^2	0.009	0.012

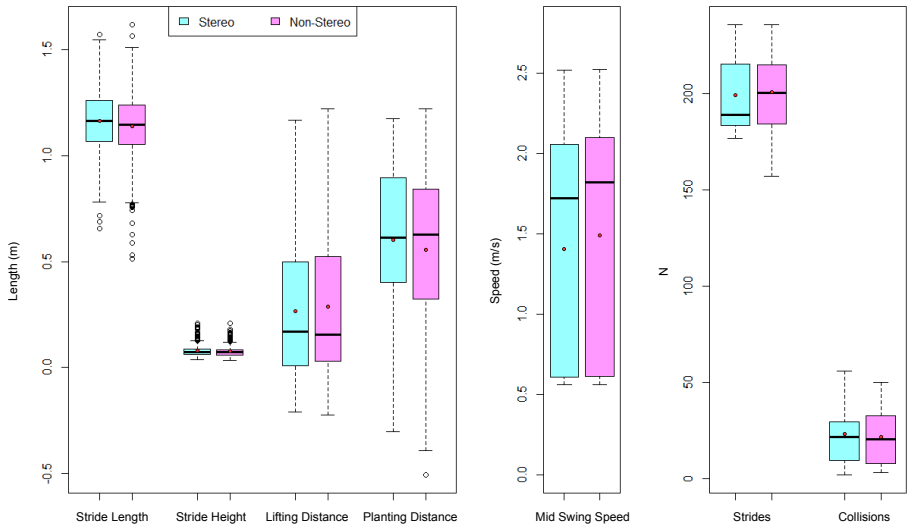


Figure 4-12: Gait Parameters on Stepping over Gaps by Viewing Condition (red dots denote mean values; the boxes of the number of strides and the number of collisions denote the data distribution of that of all walking trials of each viewing condition across participants; for other gait parameters, the boxes denote the data distribution from the gait parameters of all gait cycles that covered an gap for each viewing condition)

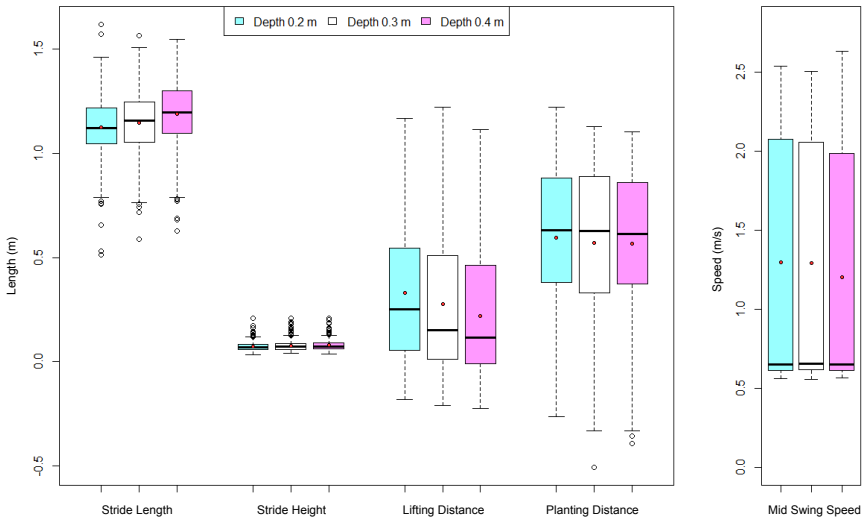


Figure 4-13: Gait Parameters on Stepping over Gaps by Depth Level (red dots denote mean values; the boxes of these gait parameters denote the data distribution from the gait parameters of all gait cycles that covered an gap for each level of gap depth)

In terms of viewing conditions, I found significant effects on stride length l_s ($F(1, 1185) = 13.01, p < 0.001, \eta_p^2 = 0.011$), stride height h_s ($F(1, 1185) = 4.87, p = 0.028, \eta_p^2 = 0.004$), foot planting distance to gaps d_p ($F(1, 1185) = 8.68, p = 0.003, \eta_p^2 = 0.007$) and mid swing speed s_f ($F(1, 1185) = 5.35, p = 0.021, \eta_p^2 = 0.004$) but not on foot lifting distance to gaps d_l ($F(1, 1185) = 1.94, p = 0.164, \eta_p^2 = 0.002$). As can be seen in Figure 4-12, stereoscopic viewing condition tended to result in larger stride height and stride length. By common sense, this was advantageous as larger stride length and stride height would help people to avoid stepping into gaps. Although the analysis on foot planting distance did not reach statistical significance, the mean value of the parameter under stereoscopic condition was generally smaller than non-stereoscopic condition, which meant that participants tried to step as close to the front edges of gaps as possible before walking over them. The result was consistent with that of Experiment 4-1. I also found that foot planting distance to the back edges of gaps was also larger under stereoscopic viewing condition than non-stereoscopic viewing condition. The result was meaningful in the sense that if we wish to safely step over a gap, a reasonable strategy is to first step as close to the front edge of the gap as possible with a foot, then make a stride to go over the gap with the other foot and plant the foot as far as possible to the other edge of the gap to avoid being tripped or trapped. The result verified that stereoscopic vision supported this strategy. I speculated that if the distance between the front edge and back edge of gaps were designed larger with a treadmill that has a longer belt, the effect on lifting distance to gaps might be significant as participants would have to step near the front

edge of the gaps more closely and accurately to make strides long enough to cover gaps. The mean value of mid swing speed s_f was lower under stereoscopic condition than non-stereoscopic condition.

Similarly, for depth levels, there were significant effects on stride length l_s ($F(2, 1185) = 29.12, p < 0.001, \eta_p^2 = 0.047$), stride height h_s ($F(2, 1185) = 6.31, p = 0.002, \eta_p^2 = 0.011$), foot lifting distance to gaps d_l ($F(2, 1185) = 17.36, p < 0.001, \eta_p^2 = 0.028$) but not on foot planting distance to gaps d_p ($F(2, 1185) = 1.27, p = 0.281, \eta_p^2 = 0.002$) and mid swing speed s_f ($F(2, 1185) = 0.09, p = 0.913, \eta_p^2 = 0.000$). Tukey's range tests revealed that there were significant differences between depth level 0.2 m and 0.4 m and between depth level 0.3 m and 0.4 m on stride length l_s; and significant differences between depth level 0.2 m and 0.3 m and between depth level 0.2 m and 0.4 m on stride height h_s and foot lifting distance to gaps d_l. Therefore, people adjusted their footsteps for gaps with different depths.

There was no interaction effect between viewing conditions and depth levels on gait parameters and there were no significant effects on number of strides n_s ($F(1, 29) = 0.25, p = 0.618, \eta_p^2 = 0.009$) and number of collisions n_c ($F(1, 29) = 0.35, p = 0.561, \eta_p^2 = 0.012$) between viewing conditions. The result on number of collisions suggested that it was equally difficult to avoid collisions with gaps under stereoscopic viewing and non-stereoscopic viewing in virtual environments.

4.5 General Discussion

Comparing the results of gait performance on stepping over obstacles and stepping over gaps, I found that stereoscopic viewing increased the number of strides significantly when stepping over obstacles but did not have a significant effect on cadence while stepping over gaps. I suspected that stepping over obstacles was a more stressful and challenging task than stepping over gaps, hence making smaller strides increased the flexibility in adjusting footsteps before stepping over obstacles. Stereopsis helped people to make smaller strides to perform more accurate movements. I also found that for both cases, mid swing speed was significantly slower under stereoscopic viewing than non-stereoscopic viewing. This probably meant that stereopsis gave better control of lower limbs, which resulted in lower mid swing speed. In addition, stereopsis shortened the foot lifting distance to the front of obstacles and gaps and increased the foot planting distance to the back of obstacles and gaps. This generally increased the chance to successfully step over obstacles or gaps, as given limits on the maximum stride length that a person can make, shortening the lifting distance to obstacles or gaps makes it more likely to plant the foot successfully after obstacles or gaps. Finally, I found that avoiding collision with obstacles or gaps was equally difficult in virtual environments under stereoscopic viewing and non-stereoscopic viewing conditions. Although people were able to make a stride with enough length and height, the trajectories of their feet may still collide with the bounding boxes of obstacles or gaps. A probable reason was that force feedback or other types of feedback, including visual or sound, were lacking in the VR system. People were not aware when their feet collided with

the bounding boxes, so it was not possible or necessary for people to make improvement on their stepping.

In addition to treadmills, other walking platforms such as the Virtuix Omni or the Cyberith Virtualizer could be integrated with the WISE. These allow people to turn and to walk with self-selected speed in VR. More complex experiment scenarios can be designed based on these platforms. A branch of locomotion techniques that does not require mechanical devices to reposition users to the centers of immersive projection-based VR systems are Walking-in-Place (WIP) techniques (Slater *et al*., 1995; Templeman *et al*., 1999; Yan *et al*., 2004; Feasel *et al*., 2008; Wendt *et al*., 2010; Williams *et al*., 2011; Wilson *et al*., 2014; Bruno *et al*., 2017). As several different WIP algorithms have been developed and they are important for practical VR approaches, it may be worthwhile to investigate how stereopsis affects walking performance with these techniques.

Matthis and Fajen (2014) found that walkers relied on visibility of the ground at least two steps ahead to locomote normally. If the visibility is less than two steps, walkers will have problems in avoiding obstacles. Their experimental approach was to project color blobs onto floor with different levels of visibility range in real-time while participants were walking. A limitation was that the projected color blobs were planar and therefore these were different from volumetric obstacles people encounter in their daily lives. For future research, I could examine the effects of occluded visual field on gait performance by masking the projected image on the display using the VR paradigm presented in this dissertation.

In this chapter, I presented two VR walking experiments to investigate the role of binocular vision in continuous walking. my results showed that stereopsis helped people to step over obstacles and gaps more accurately. I also found that stereopsis helped people to lift their feet higher for obstacles with a height of 0.3 m. Further research should investigate the threshold of the height of obstacles that enabled stereopsis to influence stride height.

To conclude, the current study suggests that providing binocular cues to VR displays is essential to design VR systems as binocular cues make stepping movements more accurate. One type of VR locomotion game, where this would be important, requires users to walk or run in virtual environments while avoiding obstacles using a locomotion interface for physical exercise or for fun, one can expect that by using a VR display with binocular cues, such gaming experience will resemble the experience in the real-world, making VR locomotion games more interesting and appealing to people.

Chapter 5
Recognition of Head Gestures and Hand Gestures and their Application for Interaction in Virtual Reality

5.1 Introduction

The goal of the research was to develop an algorithm for real-time head gesture recognition on HMDs. Such an algorithm could be useful for interaction in both virtual environments and real worlds. For instance, in virtual reality systems, users usually need to interact with avatars. To answer Yes/No questions asked by avatars, users could simply make their responses by nodding and shaking heads through a head gesture interface. One possible application is to use the interface to interact with virtual tour guides in augment reality (AR) based tours (Abate *et al*., 2011). It is also very common that a virtual reality system itself may raise questions to users and ask users to confirm or reject certain options; in this case, the user can also respond through a head gesture interface by nodding and shaking. Recently, there is a growing interest for teleoperation of robots, such as quadcopters, using head motions tracked by HMDs (Mollet and Chellali, 2008; Martins and Ventura, 2009; Higuchi *et al*., 2013; Teixeira *et al*., 2014; Pittman and LaViola, 2014). First-person views of robots are often presented directly through the display panels of HMDs. In the case of the quadcopter control, a quadcopter can be maneuvered by head spatial translations (Higuchi *et al*., 2013) or the head orientation can be used to manipulate the attitude of the quadcopter to fly it (Pittman and LaViola, 2014). Adding a head gesture interface in such applications will enable users to perform more complex operations. For example, users

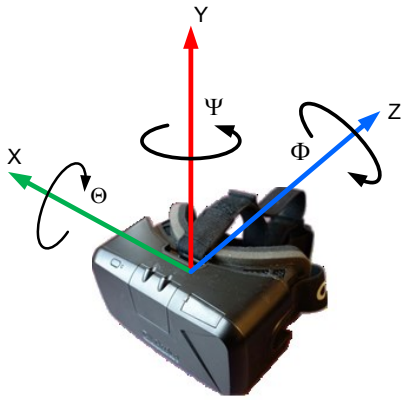


Figure 5-1: The Coordinate System of the Oculus Rift DK2

could nod their head to perform mode switching to switch flight control from auto-pilot to head motion control. Head gesture interfaces also can be applied to AR devices, such as the Google Glass and the Microsoft HoloLens. Such an interface would enable users to perform actions, such as browsing, with head rotation, head tilting, nodding and shaking without touching the glasses. A head gesture recognition method is beneficial to researchers who investigate human locomotion or driving behaviors, *etc*. In such activities, users usually have to make head movements for observing the environment around them. A head gesture recognition method can be used to count the number, the type and the duration of each head movement. These parameters may differ significantly given different experimental conditions, such as restricted field of views and complexity of the environment. Without such an approach for recognizing head gestures, researchers have to manually determine the number, the type and the duration of each head movement from the collected data of head movements. Lastly, a head recognition module also can be integrated to driver assistance systems of automobiles to monitor the driving behaviors of drivers (Kang,

2013). While the present study focused on a specific HMD - the Oculus Rift DK2, the proposed method is general and can be adapted to work with VR systems that use head tracking glasses or other types of systems in which a user's head motion is tracked by fast and accurate tracking systems.

5.2 Method

5.2.1 Head Gesture Interface

A. The Tracking System of the Oculus Rift DK2

The Oculus Rift DK2 uses a six Degree-of-Freedom (DOF) hybrid optical-inertial tracker to track a user's head motion at approximately 75 Hz. The coordinate system of the Oculus Rift DK2 is illustrated in Figure 5-1. The hybrid tracker consists of an external camera with an infrared filter to track the infrared LED array on the front and side panels of the Oculus Rift DK2 and an embedded inertial measurement unit (IMU) (LaValle *et al*., 2014). The tracking data that can be accessed through the VR software WorldViz Vizard 5.0 are position, acceleration, Euler angles and angular velocities. To recognize head gestures, I only used the angular velocities of head motions as angular velocities directly reflect whether a user's head is moving and in which direction. An advantage is that it does not assume that a user's head is in neutral position. When a user performs head gestures with their head tilted at certain angles, head gestures can still be recognized. I represented the angular velocity as a 3-D vector $\omega = (\dot{\Psi}, \dot{\Theta}, \dot{\Phi})$, where $\dot{\Psi}, \dot{\Theta}, \dot{\Phi}$ are yaw velocity, pitch velocity and roll velocity, respectively. A sequence of angular velocities can be denoted as

$W = \omega_1\omega_2 \ldots \omega_i$, where i is the index for a 3-D vector ω. Another option for monitoring head movements is to use quaternions to represent the head angular velocity but extra time will be taken to convert head angular velocities to quaternions.

B. Definition of Head Gestures

I defined nine classes of head gestures. Seven are simple gestures: Being Idle (remaining still), Rotating Left, Rotating Right, Tilting Upward, Tilting Downward, Leaning Left, Leaning Right. Two are complex head gestures: Shaking, Nodding.

The motivation behind the definition of complex gestures is that Shaking can be represented a sequence of three simple head gestures, which are Being Idle, Rotating Left and Rotating Right. Similarly, Nodding can be represented by a sequence of simple gestures: Being Idle, Tilting Upward and Tilting Downward. I associated each class of head gesture with a class label l, with $l \in \{1,2, \ldots ,9\}$.

C. Cascaded Hidden Markov Models

To recognize and classify the simple and complex head gestures, I used Cascaded Hidden Markov models. An HMM (Rabiner, 1989) is governed by the following parameters: N the number of hidden states, M the number of observation symbols and the model parameter $\lambda = (A, B, \pi)$, where A is the matrix that represents the transition probability between states, B the matrix that represents the emission probability of a symbol observed from a specific state and π the initial state probabilities. Similar to the speech recognition approach (Rabiner, 1989), I modeled each head gesture with a left-right HMM, where l is the

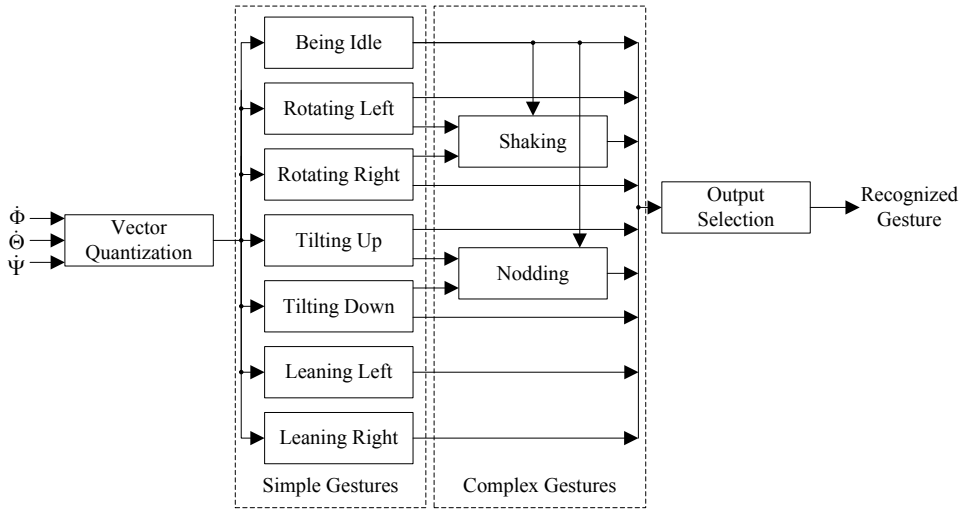


Figure 5-2: The Structure of the CHMMs for Real-time Head Gesture Recognition

class label of a head gesture associated with the HMM λ. In a left-right HMM, only transitions between adjacent states from left to right and transitions from a state to itself are allowed. This makes it possible for the proposed structure to be completely pipelined. A set of trained HMMs for the nine classes of head gestures can be represented as Λ, $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_9\}$. The HMMs I used were discrete HMMs and discrete HMMs only accept discrete observation symbols as inputs. Thus, given an sequence of sampled angular velocities W, before feeding the sequence W into an HMM λ_l, I used the K-Means algorithm during training step and the minimum distance classifier (Lin and Venetsanopoulos, 1993) during testing step as the vector quantization (VQ) procedure to quantize the sequence of angular velocities W into an observation sequence S that consists of discrete observation symbols with $S = O_1 O_2 \ldots O_i$, where i is the index of the observation symbol O in the sequence S. To predict how likely it is a sequence S belongs to a certain class l of a

head gesture, I used a set of trained HMMs Λ and the forward procedure of the HMM to calculate the posterior probabilities P_a for all HMMs, with $P_a = \{P(S \mid \lambda_1), P(S \mid \lambda_2), \ldots, P(S \mid \lambda_9)\}$. An output selection procedure resolves the class label l of the given observation sequence S from the posterior probabilities P_a. As simple gestures require much less data to give a reliable estimate than complex gestures do, to efficiently recognize simple and complex head gestures, I organized the set of HMMs Λ into the CHMM structure. The structure has two dedicated layers for recognizing simple gestures and complex gestures respectively. For training and testing the CHMM, two HMM algorithms were used. One is the Baum–Welch algorithm, which was used to train a left-right HMM λ_l. The other is the forward procedure of the HMM, which calculates the posterior probability $P(S \mid \lambda_l)$ of an observation sequence S given an HMM λ_l. These two algorithms are available in Matlab 2014a as hmmtrain() and hmmdecode(). Detailed descriptions of the algorithms can be found in the work by Rabiner (1989).

In Figure 5-2, I present the proposed CHMM structure. Here I describe the real-time operation of the CHMM and leave the explanation of the training and testing procedures to Section 5-1. During real-time operation, the system continuously reads angular velocity ω_i for processing. A sampled angular velocity ω_i is given into the vector quantization module and the vector quantization module produces an observation symbol O_i based on the Euclidean distance between cluster centers C_j and the vector ω_i using the minimum distance classifier (Lin and Venetsanopoulos, 1993): the index j of cluster center C_j

that gives the shortest Euclidean distance is assigned as the observation symbol to the vector ω_i:

$$O_i = \underset{j}{\operatorname{argmin}} \left(\left\| C_j - \omega_i \right\|_2 \right) \tag{5.1}$$

where O_i is the assigned observation symbol and cluster centers C_j were obtained by K-Means during training. The observation symbol O_i generated by the vector quantization module is buffered using an array until the number of observation symbols O_i reaches the length L_B. When the number of observation symbol O_i reaches L_B, the sequence of observation symbols S is sent to the modules classifying simple gestures to calculate the posterior probabilities P_s, with $P_s = \{P(S \mid \lambda_1), P(S \mid \lambda_2), \ldots, P(S \mid \lambda_7)\}$. The buffer is then immediately cleared and waits for new observation symbols O_i. If a recognized simple gesture l is in the category of Being Idle, Rotating Left, Rotating Right, Tilting Up or Tilting Down, they will be considered as observation symbols O_i' for the layer of complex gestures and will be further buffered using a queue of a length L_Q. Each time the queue performs a dequeue and an enqueue operation, the buffered sequence in the queue will be sent to the module of complex gestures to calculate the posterior probabilities P_c of complex gestures, with $P_c = \{P(S \mid \lambda_8), P(S \mid \lambda_9)\}$. I empirically set $L_B = 10$ and $L_Q = 10$ to make the array contain 0.13 s of data and the queue contain 1.3 s of data at the sampling rate of 75 Hz. I found such choice of values gave a relatively fast response for recognizing simple gestures and reliable length of data for recognizing complex gestures. The last step is to resolve the class label l of the head gesture based on the calculated posterior probabilities P_s and P_c.

The posterior probabilities of simple gestures P_s and the posterior probabilities of complex gestures P_c are not directly comparable as a complex gesture consists of symbols represented by simple gestures. To solve this problem, I proposed an output selection procedure to estimate the class label $\hat{l}$ from the posterior probabilities of simple gestures P_s and complex gestures P_c using the function:

$$\hat{l} = \begin{cases} \underset{l}{\text{argmax}}(P_c) & if\ P(S \mid \lambda_8) > \tau_n\ or \\ & if\ P(S \mid \lambda_9) > \tau_s \\ \underset{l}{\text{argmax}}(P_s) & otherwise \end{cases} \tag{5.2}$$

where τ_n and τ_s are the thresholds that indicate a complex gesture shaking or nodding may exist if either the posterior probability $P(S \mid \lambda_8)$ or $P(S \mid \lambda_9)$ is larger than their corresponding thresholds τ_n or τ_s; l is the class label associated with a HMM λ of a specific gesture and $\hat{l}$ is the estimated class label.

5.2.2 Hand Gesture Interface

Marin *et al*. (2016) proposed a set of robust features for recognizing static hand gestures with the hand skeletons tracked by the Leap Motion sensor. The specific feature descriptor selected from the set for my implementation was:

$$\begin{aligned} P_i^x &= (F_i - C) \cdot (n \times h) \\ P_i^y &= (F_i - C) \cdot h \\ P_i^z &= (F_i - C) \cdot n \end{aligned} \tag{5.3}$$

where F_i is the position of the fingertip and i is the index of a finger, C the position of the palm center, n the normal vector emanating from the palm and h the vector from the palm

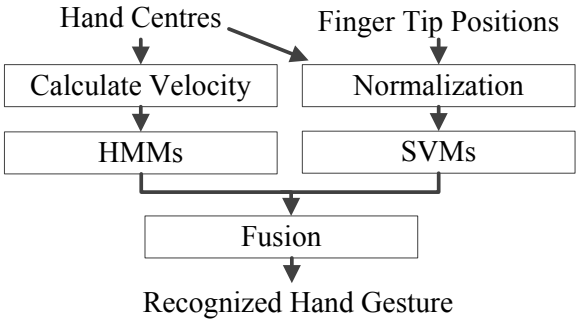


Figure 5-3: Hand Gesture Interface

center to the direction of the fingers. These parameters are directly available from the tracked hand skeleton of the Leap Motion Sensor. P_i^x, P_i^y and P_i^z are the extracted features. $\cdot$ is the dot product and $\times$ is the cross product. As pointed out by the authors, the set of equations normalize fingertip positions with respect to hand position and orientation. Fingertip angles, positions and elevations are embedded in the extracted features P_i^x, P_i^y and P_i^z. The extracted features can be used to train classifiers such as the Support Vector Machine (SVM) (Chang and Lin, 2011) to recognize static gestures.

Initially, I defined three types of hand gestures. The OK gesture represents Yes, the extended hand gesture means No. The fist gesture was also defined as the standby gesture for resting.

A problem using only the static hand gesture recognition algorithm is that it is difficult to determine whether users intend to confirm their responses as static hand gestures are continuously recognized and a response can be determined before users finishing making their intended gestures. Thus, I included two HMMs to monitor the trajectory of the

Figure 5-4: Gamepad Interface

hand velocity to detect if users are waving their hands or not. The outputs from the SVMs and the HMMs are fused by a set of rules to generate the final gesture: if the user waves a hand with an OK gesture, then the algorithm will confirm that the response from the user is Yes; similarly, if the user waves an extended hand, the response will be confirmed as No; otherwise, the algorithm considers that there are no meaningful responses given by users. Therefore, the types of hand gestures in the hand gesture interface were extended to six types, including: static OK gesture, static extended hand, static fist, waving OK gesture, waving extended hand and waving fist. The structure of the hand gesture recognition algorithm is illustrated in Figure 5-3.

An advantage of the proposed hand gesture recognition framework is that it can be further extended to recognize combinations of different static hand gestures and different shapes of hand velocity trajectories. Thus, it has the potential to deal with more complex gesture recognition scenarios.

5.2.3 Gamepad Interface

The gamepad interface (Figure 5-4) was implemented based on a Logitech Dual Action gamepad to be compared with the head gesture interface and the hand gesture interface. Specifically, users pressed button 5 on the gamepad for Yes and pressed button 6 for No.

5.3 Experiment 5-1: Training and Testing of the Head Gesture Interface

In this experiment, I trained the CHMM and evaluated its offline classification performance. As there was no publicly available head gesture dataset for the Oculus Rift DK2, I developed a custom application, using the Vizard 5.0, that can simultaneously collect and label head gestures for training and testing the proposed CHMM structure. Nineteen people (age: 20 - 38) participated in the experiment and informed consents were obtained from all participants in accordance with a protocol approved by the Human Participants Review Subcommittee at York University. In the experiment, the participants wore the Oculus Rift DK2 and sat approximately 60 cm in front of the tracking camera of the Oculus Rift DK2, which was mounted on the monitor of the host machine (Windows 7, an Intel i7 2.8 GHz. CPU, 4 GB memory and an AMD Radeon HD 6850 graphics card). There were nine types (classes) of head gestures that needed to be collected. For each type of head gesture, the researcher pressed the corresponding button on the control panel (visible only to the researcher on the computer monitor) of the custom application and a prompt (visible to both participants on the HMD and the researcher on the computer monitor) indicating the type

Table 5-1: The Average Accuracy of the Simple Gesture Layer from a Training Session (Unit: Percentage, M: The Number of Discrete Symbols in HMMs, N: The Number of Hidden States in HMMs; The Highest Average Accuracy is in Bold and Shaded)

		N				
		2	3	4	5	6
	7	87.6	85.7	87.6	86.1	88.1
	8	90.1	90.1	92.7	90.3	90.4
	9	95.9	95.9	95.9	95.9	96.2
	10	92.6	91.6	97.4	94.0	96.1
	11	95.2	97.0	97.0	97.4	98.4
	12	97.2	96.2	97.3	98.6	98.7
	13	97.5	98.8	99.0	98.7	98.7
	14	98.6	98.7	98.4	98.7	98.3
M	15	97.0	99.0	99.1	98.8	98.7
	16	98.7	98.9	98.8	99.0	99.1
	17	98.6	**99.2**	99.0	99.0	99.1
	18	98.9	98.4	99.1	99.0	98.9
	19	97.6	98.9	99.0	98.9	98.9
	20	98.7	99.1	97.3	97.5	99.0
	21	98.8	97.4	97.5	99.0	98.9
	22	97.6	99.0	97.2	98.8	98.7
	23	97.4	97.5	97.3	97.8	97.4
	24	97.5	97.6	97.7	97.4	97.0
	25	97.4	97.5	97.8	97.6	97.2

of head gesture that a participant needed to perform was shown in the view of the HMD. A countdown timer was also started at the same time to count from two to zero by seconds.

A participant was expected to complete the head gesture within 2 seconds with their preferred head movement speed. Labeling of the head gesture was done at the same time by the custom application within the 2-second interval. I collected each type of head gesture twice for each participant. In total, I had 342 head gesture samples from nineteen participants. The training and testing procedures were fully automated and performed using Matlab 2014a. I used multi-class precision ($Precision_M$), multi-class recall ($Recall_M$) and

Table 5-2: Head Gesture Recognition Latencies (Unit: s)

	RL	RR	TU	TD	LL	LR	S	N
P1	0.213	0.253	0.173	0.120	0.200	0.240	0.720	0.653
P2	0.173	0.173	0.147	0.120	0.107	0.160	0.747	0.733
P3	0.107	0.093	0.133	0.173	0.227	0.253	0.813	0.667
P4	0.267	0.080	0.093	0.200	0.253	0.120	0.773	0.680
P5	0.280	0.280	0.173	0.147	0.080	0.173	0.667	0.733
P6	0.267	0.133	0.160	0.200	0.120	0.120	0.693	0.693
Mean	0.218	0.169	0.147	0.160	0.164	0.178	0.736	0.693
Std	0.068	0.083	0.030	0.037	0.071	0.058	0.054	0.034

the average accuracy (Sokolova and Lapalme, 2009) as the metrics to evaluate the performance of training and testing. I divided the training of the CHMM into three phases that trained the modules of vector quantization, simple gesture layer and complex gesture layer separately. Given the collected head gesture dataset from nineteen participants, I directly ran the K-means algorithm to quantize each angular velocity vector ω_i into an observation symbol O_i. The cluster centers C_j obtained from K-Means training were stored for the testing step for the vector quantization module. The cluster number K of K-Means and the number of observation symbols M of HMMs were equated ($M = K$). Seven types of observation symbols, with $M \geq 7$, were needed to represent seven simple gestures. The head gesture dataset was then divided in halves into a training set and a testing set. For each sequence of head gestures other than being idle, I removed redundant symbols that indicate a user's head is remaining still as the redundant symbols were not useful for training.

The second step was to train the layer of simple gestures and evaluate its classification performance. To represent a simple head gesture, such as rotating left, at least two hidden states N are needed, with $N \geq 2$. The Baum–Welch algorithm was used for training. Since I chose $L_B = 10$ for real-time evaluation, I partitioned each sequence in the

training set into short sequences of length 10 and used the short sequences to train its each associated HMM λ_l, $l \in \{1, 2, \ldots, 7\}$, to obtain the parameters of the transition matrix A and the emission matrix B. The initial state probabilities π were not considered as a head gesture always starts with the state of the head being remaining still. The information can be learned during training and stored in the emission matrix B. There were two tunable parameters: N the number of hidden states in the model, M the number of observation symbols in an HMM and I knew that $N \geq 2$ and $M \geq 7$. To obtain the best classification performance, I wished to find the optimal values of N and M that maximize the average accuracy for the layer of simple gestures, with smallest M and N possible. The smaller N and M are, fewer additions and multiplications are involved in the forward procedure of the HMM; hence the faster real-time performance for head gesture recognition. The upper bounds for N and M were set to 6 and 25 empirically. The evaluation of the classification performance of the simple gesture layer was conducted after each training cycle with a combination of N and M. I partitioned each sequence in the testing set into the short sequences of length 10 and used the forward procedure to calculate the posterior probabilities P_s based on short sequences. The class label $\hat{l}$ of a simple head gesture was estimated using the equation:

$$\hat{l} = \underset{l}{\operatorname{argmax}}(P_s) \tag{5.4}$$

Since the K-means is initialized randomly, the training results may differ even if I run the same algorithm on the same training set. Thus, I ran the training procedure for the

layer of simple gestures for five different sessions. In each session, the training was performed with different combinations of N and M such that $2 \leq N \leq 6, N \in \mathbb{Z}$ and $7 \leq M \leq 25, M \in \mathbb{Z}$. The highest average accuracy I was able to obtain from a specific training session was 99.2% when $N = 3$ and $M = 17$ (Table 5-1) and the corresponding multi-class precision and multi-class recall were 97.9% and 96.6%, respectively.

To train the complex gesture layer and evaluate the classification performance, I first partitioned the sequence of nodding and shaking in the head gesture datasets into short sequences of length 10 (since $L_Q = 10$). I then used the trained HMMs $\Lambda_s = \{\lambda_1, \lambda_2, \ldots, \lambda_7\}$ of simple gestures to classify the short sequences of complex gestures into observation symbols O_i' that consists of classified simple gestures. Specifically, the classification was done by first using the forward procedure of the HMM to calculate the posterior probabilities of each short sequence of complex gestures and assigning the short sequence with the class label of the simple gesture with the highest posterior probability using equation (5.4). As a complex gesture is represented by three simple gestures, I set the number of observation symbols $M = 3$. The number of hidden states was set as $N = 3$, which was same as that of the simple gesture layer. I then used the Baum–Welch algorithm to train the layer of complex gestures and obtained the parameters for the transition matrix A and the emission matrix B for HMMs of complex gestures $\Lambda_c = \{\lambda_8, \lambda_9\}$. As with the training of the simple gesture layer, the initial state probabilities π were not considered. To determine the thresholds τ_n and τ_s, I used the forward procedure and the HMMs $\Lambda_c = \{\lambda_8, \lambda_9\}$ to calculate the posterior probabilities P_c of all sequences of complex head gestures in the training

set. I selected the smallest values as the thresholds τ_n and τ_s for shaking and nodding, respectively, with τ_n = -6.93 and τ_s = -7.54 (on a logarithmic scale). The last step was to test the layer of complex gestures. I calculated the posterior probability P_c , $P_c = \{P(S \mid \lambda_8), P(S \mid \lambda_9)\}$, of each complex gesture sequence in the testing set, compared the posterior probabilities P_c with the thresholds τ_n and τ_s I obtained during the training procedure and estimated the class label of each sequence using the equation:

$$\hat{l} = \begin{cases} \underset{l}{\text{argmax}}(P_c) & if\ P(S \mid \lambda_8) > \tau_n\ or \\ & if\ P(S \mid \lambda_9) > \tau_s \\ -1 & otherwise \end{cases} \tag{5.5}$$

where -1 will be given as an invalid class label when both $P(S \mid \lambda_8)$ and $P(S \mid \lambda_9)$ were lower than their corresponding thresholds τ_n and τ_s.

The multi-class precision, the multi-class recall and the average accuracy were 100%, 96.4% and 98.5%, respectively, for the layer of complex gestures.

5.4 Experiment 5-2: Estimating the Latency of the Head Gesture Interface to Recognize Head Gestures

For real-time evaluation of the latency of the head gesture recognition framework, I implemented the proposed CHMM structure and the forward procedure of the HMM using python 2.7 in Vizard 5.0. My goal was to estimate the latency for the algorithm to recognize a head gesture. In practice, I found it necessary to further tune the parameters τ_n and τ_s such that $\tau_n = -5$ and $\tau_s = -4$. This helped the CHMM avoid confusing fast head rotation and tilting with Shaking and Nodding during real-time recognition. Each computation

cycle of the CHMM takes approximately 1 ms and thus the proposed approach meets the target real-time requirement of completing a computation cycle within 13 ms at the sampling rate of 75 Hz. Nine people (age: 22 - 31) participated in the experiment and none had participated in Experiment 5-1. Informed consents were obtained from all participants in accordance with a protocol approved by the Human Participants Review Subcommittee at York University. In Experiment 5-2, participants also wore the Oculus Rift DK2 and sat 60 cm in front of the tracking camera of the Oculus Rift DK2. Participants were asked to perform the following head gesture sequence with their preferred speed three times from the initial neutral position with a head gesture of **Being Idle (BI)**:

- **Rotating Left (RL);**
- **Move Back to Neutral Position;**
- **Rotating Right (RR);**
- **Move Back to Neutral Position;**
- **Tilting Upward (TU);**
- **Move Back to Neutral Position;**
- **Tilting Downward (TD);**
- **Move Back to Neutral Position;**
- **Leaning Left (LL);**
- **Move Back to Neutral Position;**
- **Leaning Right (LR);**
- **Move Back to Neutral Position;**

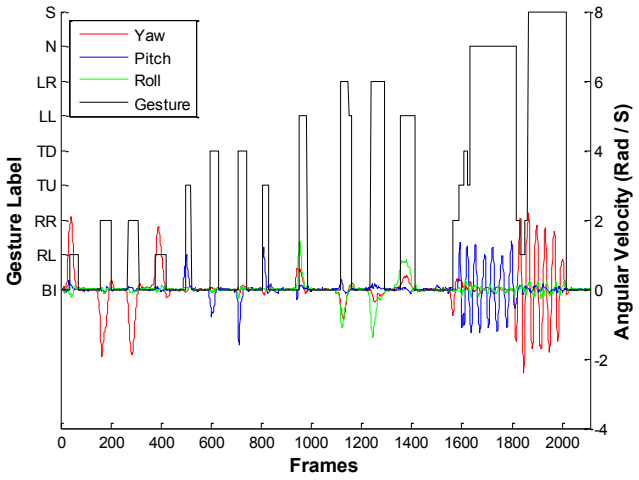


Figure 5-5: An Exemplar Real-time Recognition Result from a Participant

- **Nodding (N);**
- **Move Back to Neutral Position;**
- **Shaking (S);**
- **Move Back to Neutral Position;**

The estimated head gesture labels and head angular velocities during real-time operation were recorded for latency analysis (Figure 5-5). I defined the latency for head gestures, except Being Idle, as the time interval between the index of the frame i_{init} when the Euclidean norm of a user's head angular velocity $\|\omega\|$ is equal to or higher than a threshold $\omega_{init} = 0.1$ rad / s, to the index of the frame $i_{trigger}$ at which a head gesture was recognized. Then the latency t_L can be estimated as:

$$t_L = \frac{i_{trigger} - i_{init}}{f_s} \tag{5.6}$$

where f_s is the sampling frequency of the Oculus Rift DK2 and $f_s = 75$ Hz.

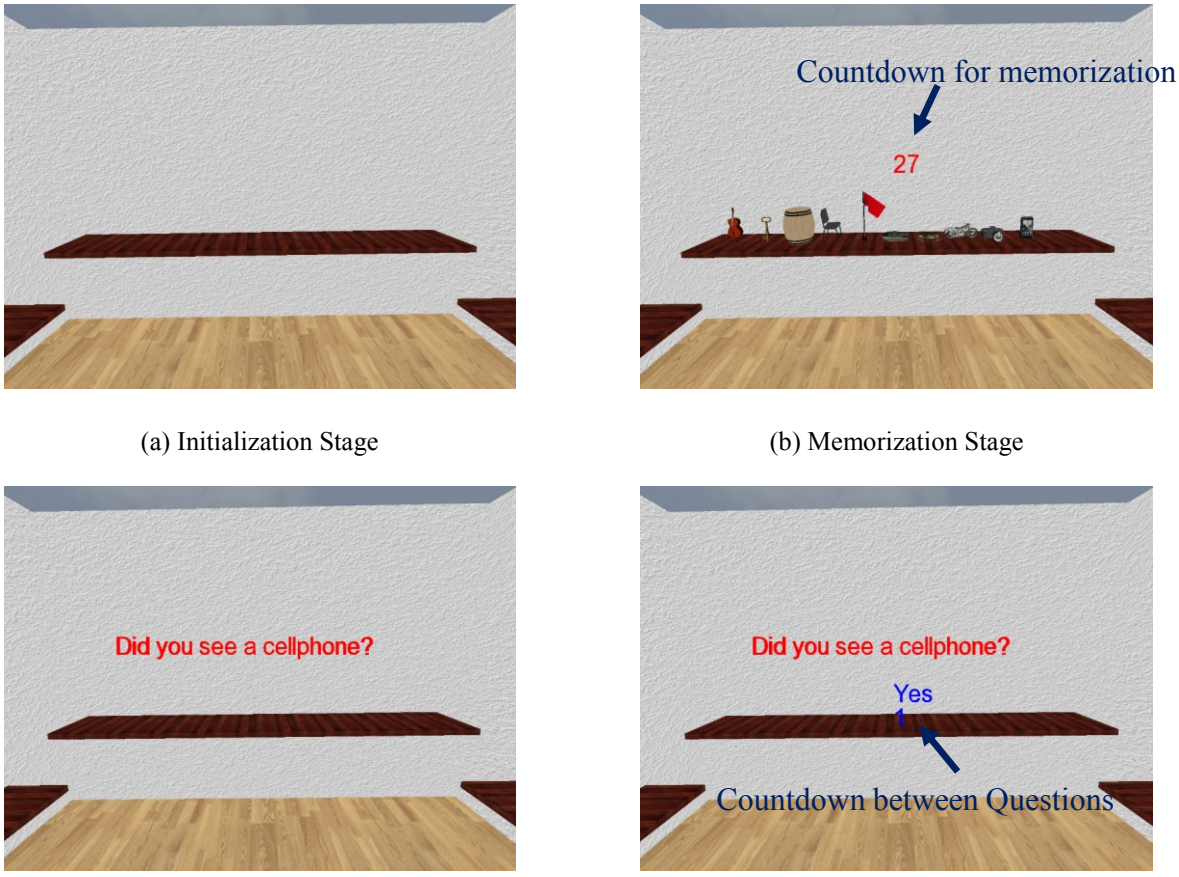
Three participants were unable to follow the gesture sequence given above as they did not remember the sequence they needed to perform. Thus, I only used the data of the remaining six participants (age: 22 - 31) and the researcher selected one specific sequence among the three that a participant performed to estimate the latency. The selected sequence was the clearest pattern compared with that of other two sequences. The other two sequences might not have the states of BI between other recognized head gesture states due to users' continuous head movements, which made it more difficult to estimate the latency. In Table 5-2, I present the result of the estimated head gesture recognition latencies of six participants (P1 – P6). The mean value and the standard deviation of the latencies of all participants were also calculated. I found that for simple gestures, the latency had a mean value of 0.17 s and for complex gestures the latency had a mean value of 0.71 s.

5.5 Experiment 5-3: Training and Testing of Hand Gesture Interface

I collected hand gesture samples from twelve participants (age: 20 - 33) for the six types of gestures. Informed consent was obtained from all twelve participants in accordance with a protocol approved by the Human Participants Review Subcommittee at York University. Each participant was asked to perform four sessions of data collection. In each session, a participant was asked to perform the six types of hand gestures separately and each type of hand gesture was recorded for four seconds.

Table 5-3: The User Interface Questionnaire

1.	The interface is easy to learn.
2.	The interface is easy to use.
3.	The interface is natural and intuitive to use.
4.	The interface helps make the task fun.
5.	Using the interface is tiring.
6.	The interface helps me respond quickly.
7.	The interface helps me make accurate responses.



(a) Initialization Stage

(b) Memorization Stage

(c) Question Stage

(d) Response Made

Figure 5-6: Experiment Stages

The features P_i^x, P_i^y and P_i^z extracted from the collected samples were used to train three SVMs using the one against one approach (Hsu and Lin, 2002). To recognize a gesture during real-time operation, a voting strategy was used, meaning that the type of gesture

that received the highest number of the votes given by SVMs was the winner. The average accuracy for recognizing static hand gestures was 99.6%. To handle moving hands, Two HMMs were trained using hand velocity data calculated from the hand centers. Specifically, one HMM was trained using hand velocities of static gestures while the other was trained using hand velocities while hands were waving. During operation, the HMM that gives the highest probability was chosen as the output. The average accuracy for recognizing a waving hand is 98.1%. The theoretical average accuracy for recognizing dynamic hand gestures was 97.7% by multiplying the average accuracy for recognizing static hand gestures (99.6%) and the average accuracy for detecting moving hands (98.1%).

5.6 Experiment 5-4: Comparing Head Gesture, Hand Gesture and Gamepad Interfaces for Answering Yes/No Questions in Virtual Environments

5.6.1 Introduction

The goal of the experiment was to evaluate and compare performance and user preference of the head gesture interface, the hand gesture interface and the gamepad interface for answering Yes/No questions in virtual environments. To achieve the goal, a memorization task was designed. The task asked participants to memorize the objects presented in a virtual room with a 30-s exposure period. Then, these objects were removed, and participants were asked whether they saw a specific object by answering Yes or No through a given interface. In Figure 5-6 (a) – (c), I show the three stages of the experiment, including ini-

tialization stage, memorization stage and question stage. When a participant made a response, a confirmation (Yes/No) was prompted as shown in Figure 5-6 (d). The experiment application and the algorithms for three interfaces were implemented in the Worldviz Vizard 5.0 using Python 2.7.

5.6.2 Metrics

The metrics consisted of objective measures and subjective measures. The objective measures were: Response Time, which is the interval between when a question was prompted and when a response was made and Real-Time Accuracy, which is the percentage of the objects that were correctly classified as present or not during real-time operation. The objective measures were applied on the recorded experiment data to extract the corresponding parameters. The subjective measures were: Ease-to-Learn, Ease-to-Use, Natural-to-Use, Fun, Tiredness, Responsiveness and Subjective Accuracy.

The subjective measures were evaluated using a user interface questionnaire modified from the one by Nabiyouni *et al*. (2015). The items given in the questionnaire are shown in Table 5-3. The seven-point Likert scale (from Strongly Disagree to Strongly Agree) was used to rate each factor.

5.6.3 Participants

Informed consent was obtained from all twelve participants (age: 20 - 38) in accordance with a protocol approved by the Human Participants Review Subcommittee at York University. Twelve participants were divided into six groups. To order the testing of the three

interfaces, each group covered a permutation of the gesture interfaces. Thus, six groups covered all six permutations of the three gesture interfaces. For each interface, four trials were conducted. The first trial was for training and was not considered for the data analysis while the remaining three trials were the actual experiments.

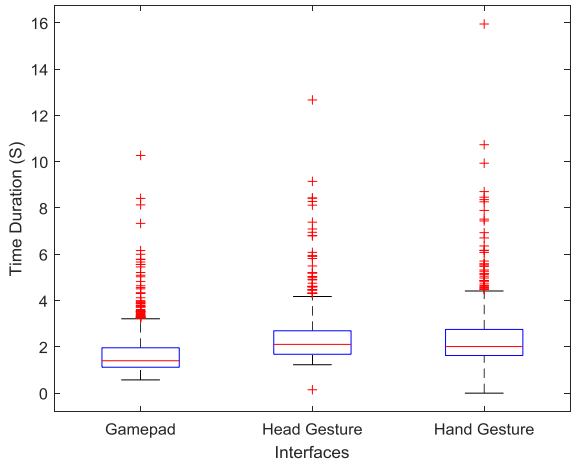


Figure 5-7: Response Time

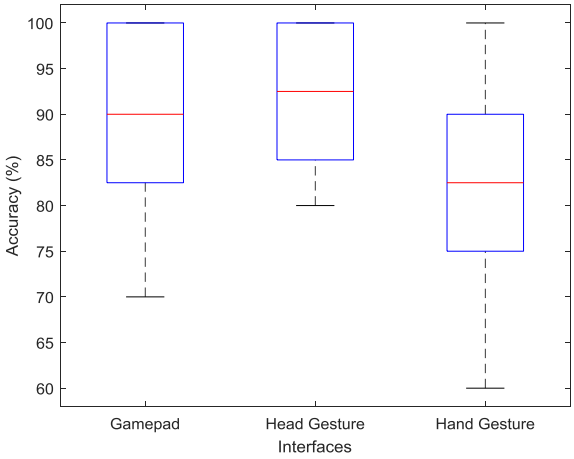


Figure 5-8: Real-Time Accuracy

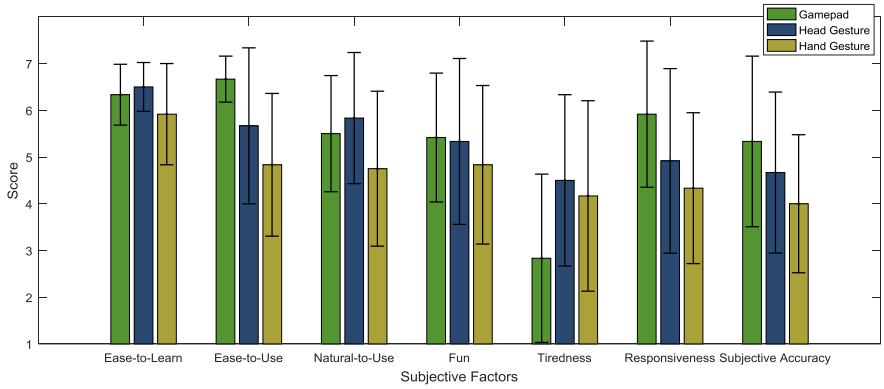


Figure 5-9: Subjective Measures

5.6.4 Procedure

During experiment sessions, participants wore the Oculus Rift DK2 and sat 60 cm in front of the computer monitor, on which the tracking camera of the Oculus Rift DK2 was mounted. The Leap Motion sensor was attached onto a stand and was placed 30 cm in front of the participants. At the beginning of a trial, a participant was exposed to a virtual room, facing a bench placed in the front of the room. After the researcher pressed the start button, twenty objects were randomly selected from a list of thirty objects. The list consisted of everyday objects, including a camera, a cellphone and a chair, *etc*. The 3-D models of the objects were obtained from www.turbosquid.com. Among the selected objects, ten were placed onto the bench with a random order and participants were given thirty seconds to memorize the presence of these objects. Another ten objects were not presented in the room and were only used for generating questions. After the thirty-second memorization period, the presented objects were removed from the room. Participants were sequentially asked

about the existence of the twenty objects with a random sequence. Questions had the form "Did you see a cellphone?" (Figure 5-6 (c)). Each time the participant made a response through the given interface, a three-second waiting period was introduced before the next question was prompted. The object names, the timestamps when the questions were prompted, the timestamps when the responses were made, the actual existence of objects and the responses of the participants were recorded for data analysis. After participants completed four trials for a given interface, they were asked to complete the user interface questionnaire to evaluate the interface they used.

5.6.5 Results

I performed the data analysis in Matlab 2016a. One-way repeated-measures ANOVA analyses were applied on each factor of the objective measures and subjective measures to reveal whether there were significant effects between the types of interfaces. Post-hoc pairwise comparisons were performed using Tukey's range tests. Effect sizes were reported using partial eta squared η_p^2.

I found a significant effect on Response Time ($F(2,22) = 50.84, p < 0.001, \eta_p^2 = 0.822$). On average, the head gesture interface had the highest response time and the gamepad interface had the lowest while the hand gesture was in the middle. A Tukey's range test confirmed that the gamepad interface was significantly faster than the head gesture interface and the hand gesture interface in terms of response time and there was no significant difference between the head gesture interface and the hand gesture interface. The results (Figure 5-7) were expected since the head gesture interface required

nodding or shaking for at least one cycle. This typically took longer time than pressing a button on the gamepad or waving hands in front of the Leap Motion sensor. I also found a significant effect on Real-Time Accuracy ($F(2,22) = 16.70, p < 0.001, \eta_p^2 = 0.604$). A Tukey's range test showed that the hand gesture interface was significantly less accurate than that of the gamepad interface and the head gesture interface. The objective accuracies for three interfaces are shown in Figure 5-8. The factor is primarily determined by the accuracy of memorization of the objects, the control of the interfaces and the recognition performance of the interfaces. Although, in theory, the gamepad interface should have the best recognition performance as Yes/No are recognized by two buttons, I found the head gesture had a slightly higher real-time accuracy than that of the gamepad interface assuming that the memorization of objects by participants across three interfaces was the same. This suggested that using the head gesture interface was less error-prone than the gamepad

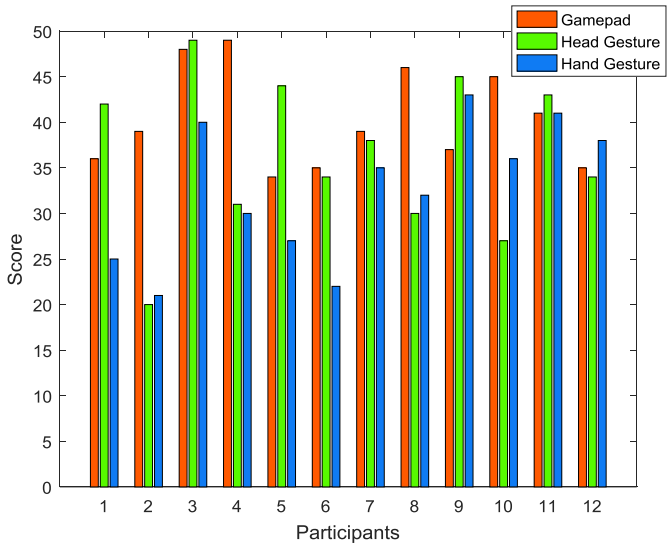


Figure 5-10: Total Score

interface. The hand gesture interface had the lowest real-time accuracy, which suggested that using hand gesture interface might introduce more errors into responses.

For subjective measures (Figure 5-9, error bars denote standard deviation), the gamepad interface was rated better than other interfaces in terms of Ease-to-Use, Fun, Tiredness, Responsiveness and Subjective Accuracy while the head gesture was rated slightly higher for Ease-to-Learn and Natural-to-Use. The hand gesture interface was not preferred for any measure except Tiredness as the head gesture interface was considered as the most tiring interface. I found a significant effect on Ease-to-Use ($F(2,22) = 7.00$, $p = 0.004$, $\eta_p^2 = 0.389$) and a Tukey's range test showed that the gamepad was significantly easier to use than the hand gesture interface. A significant effect was also found on Tiredness ($F(2,22) = 4.22$, $p = 0.028$, $\eta_p^2 = 0.277$) and a Tukey's range test showed that the gamepad interface was significantly less tiring than the head gesture interface. However, I did not find significant effects on other measures: Ease-to-Learn ($F(2,22) = 2.27$, $p = 0.13$, $\eta_p^2 = 0.171$), Natural-to-Use ($F(2,22) = 2.18$, $p = 0.14$, $\eta_p^2 = 0.165$), Fun ($F(2,22) = 0.65$, $p = 0.53$, $\eta_p^2 = 0.056$), Responsiveness ($F(2,22) = 2.89$, $p = 0.08$, $\eta_p^2 = 0.208$) and Subjective Accuracy ($F(2,22) = 2.84$, $p = 0.08$, $\eta_p^2 = 0.208$).

One interesting finding was that Responsiveness and Subjective Accuracy in the subjective measures did not agree with Response Time and Real-Time Accuracy in the objective measures respectively. For example, although subjectively participants indicated

that the gamepad interface was more accurate than the head gesture interface, this was not the case when the accuracy was assessed objectively. Similarly, the head gesture interface took the longest time for making responses on average in the objective measure, but participants indicated that the hand gesture interface was less responsive than the head gesture.

The total scores of each interface rated by all participants were produced by summing the subjective scores and are shown in Figure 5-10. The score for the factor Tiredness was inverted (Strongly Agree received one point and Strongly Disagree received seven points) to indicate how positive participants' attitudes were towards Tiredness. By total score, I found that the gamepad interface was preferred by six participants, while the head gesture interface was preferred by five participants. Only one participant opted for the hand gesture interface. Three paired t-tests were performed to further test the preference of these three interfaces in terms of total scores. Effect sizes were reported using Cohen's d_{av} (Lakens, 2013). I found that for the gamepad interface *vs*. the hand gesture interface, there was a significant difference ($t(11) = 3.427, p = 0.006,\ d_{av} = 1.214$), which showed that the gamepad interface was much preferred than the hand gesture interface. However, I did not find a significant difference between the gamepad interface *vs*. the head gesture interface ($t(11) = 1.253, p = 0.236,\ d_{av} = 0.562$) and between the head gesture interface *vs.* the hand gesture interface ($t(11) =\ 1.655, p = 0.126, d_{av} = 0.487$).

5.7 Discussion

Gamepads or other hand-held devices are traditional interfaces for people to play games and have a long history for interaction in VR systems, console gaming and PC gaming. For example, joysticks have been used for flying in virtual environments as a method for locomotion (Robinett and Holloway, 1992). Because of people's familiarity and previous experience with these devices, it is possible that even when new interfaces appear, they would still prefer these traditional devices as such devices may be more reliable. In addition, hand-held devices are more familiar and would not take extra effort for people to learn how to use them. In Figure 5-10, six participants preferred the gamepad interface to other two interfaces. This showed that gamepads or hand-held devices are still important devices for VR interactions.

Responding Yes/No through head nodding and shaking is a natural means of the interaction between people in the real-world. In Figure 5-9, the rating of Natural-to-Use was higher than that of other interfaces. Perhaps if would be particularly natural for interaction with agents or avatars. Similarly, the interface was rated easier to use than other interfaces. As has been discussed, the primary problem with the interface is the heaviness of the HMD, which probably made people consider the head gesture interface the most tiring one. I expect that by using an HMD or tracking glasses with lower weight or using computer vision systems for tracking, the tiredness for using the interface would be lowered. But given tiredness as the primary limitation, the interface was still preferred by five participants.

The hand gesture interface was only preferred by one participant probably because the definition of Yes/No using a waving OK gesture and a waving extended hand was not natural or unfamiliar to participants. To make a response, the hand of a participant needed to make a two-step movement. First, they needed to make an OK gesture or extend their hands. Then, then they needed to wave their hands to confirm their responses. It was obvious that more efforts were required when using the hand gesture interface than other two interfaces, which required only a one-step movement, such as pressing a button or shaking their heads. As shown in Figure 5-9, the hand gesture was the most difficult to learn and most difficult to use. The factors Fun, Responsiveness and Subjective Accuracy were also lower than other two interfaces. It was only considered better than the head gesture interface in terms of Tiredness. I expect that by defining better gestures for Yes and No for the hand gesture interface, people may have more positive attitudes towards the interface.

Another option to implement the functionality to answer Yes/No questions in VR systems is to use speech recognition algorithms as interfaces to recognize people's voice. The performance and user preference of the speech recognition interface also can be studied and compared to the motion-based interfaces presented in this study. In practice, I can also design a multi-modal interface that integrates the head gesture interface, the hand gesture interface, the gamepad interface and the speech recognition interface into a single system and let users choose their preferred interface during actual usage.

Limitations of the experiment design of the memorization task were that extra time was taken for participants to recall the objects they memorized when responding to Yes/No

questions using a given interface; and the ability of the participants to memorize the given objects might also affect the results of the real-time accuracy.

In this chapter, I first presented a CHMM structure for real-time head gesture recognition. The proposed structure is scalable and modules for other types of gestures can be added or the existing modules can be removed based on the application needs. A distinct advantage of the proposed pipelined structure is that the structure can be more easily implemented on Field Programmable Gate Arrays (FPGAs) and Application-specific Integrated Circuits (ASICs) compared with the cyclic structure described by Terven *et al*. (2014), as a cyclic structure is iterative and non-deterministic. Such modules can be integrated into head wearable devices, such as the Google Glass and the Microsoft HoloLens, as a dedicated module for fast head gesture recognition. A limitation with the user study based real-time evaluation is that it is impossible to ask the user to perform head motions with precise velocity and duration. Thus, the real-time classification performance of the proposed approach needs to be further evaluated with a robotic head since the velocity and the duration of a head gesture performed by a robotic head can be easily controlled by programming. This will enable me to compare the timings of head movements with that of the real-time classification results and determine the classification performance.

I also proposed to use the head gesture interface and the hand gesture interface to answer Yes/No questions in virtual environments. I evaluated their performance and user preference through a memorization task and compared them to the traditional gamepad

interface. I showed that the head gesture interface was comparable to the gamepad interface. As adding the head gesture interface to a VR system usually does not require additional tracking devices, I suggest adding the head gesture interface to VR systems that require users to answer Yes/No questions. I believe that interaction techniques using head gestures and hand gestures in VR systems is still underexplored. Thus, the utility of these interfaces in VR systems is worth further investigation.

Chapter 6
The Effects of Visual and Control Latency in a Head Motion Controlled Quadcopter

6.1 Introduction

Recently, there has been growing interest in remotely operating robots and aerial vehicles using head motion tracked by an HMD. A typical teleoperation approach for such vehicles maps the tracked head orientation by an HMD to the attitude of the vehicles for maneuvering it and first-person views from the perspective of the vehicles are usually captured by onboard cameras and presented onto the display panels of HMDs (Martins and Ventura, 2009; Higuchi *et al*., 2013; Pittman and LaViola, 2014; Teixeira *et al*., 2014). Such settings give users an egocentric, immersive and intuitive way of teleoperation compared with conventional control methods using hand-held devices, such as a joystick. However, a major difference between head motion control methods and conventional control methods is that, in the former case, head motion is coupled with visual updates. As the motion of a vehicle is constrained by its dynamics, appreciable visual and control latency always exists between the issue of control commands by head movements and the visual feedback received at the completion of the attitude adjustment. This causes a discrepancy between the intended motion, the vestibular cue and the visual cue and may potentially result in simulator sickness. In most VR applications, the major source of latency in HMDs comes from the motion-to-photon latency. However, the motion-to-photon (end-to-end) latency (Iribe, 2013) in HMDs, such as the Oculus Rift DK2, is estimated to be only a few milliseconds

(Raaen and Kjellmo, 2015; Kijima and Miyajima, 2016). This is minimal compared to the latency introduced by dynamics in vehicles, which is dependent on the hardware and the controller used. Additional latency may be introduced if the head motion of the user is not properly mapped to the motion of vehicles. An example is the threshold technique, in which the vehicle starts to move or stop when Euler angles of a user's head pass certain thresholds, or the movement of the vehicle is triggered by certain gestures. I consider this as a type of discrete ON/OFF control input. A more responsive approach is to continuously map head motions to the motion of the vehicle so that the latency introduced in the motion mapping is minimal. For example, to fly a quadcopter, head orientation can be mapped to the tilt of the quadcopter and the quadcopter moves when it is tilted. I refer this type of operation as continuous inputs. Other sources of latency in head motion controlled vehicles include network transmission (Allison *et al*., 2004) and computations, but these were not considered in the present study as I focused on visual and control latency, which dominates in the envisioned scenarios. There has been no previous research on how visual and control latency introduced by dynamics in head motion controlled vehicles affect a user's performance and well-being. A common upper bound for tolerable latency in VR systems is usually taken as 20 ms (Allison *et al*., 2001) but visual and control latency introduced by dynamics is usually much higher. Thus, it is uncertain whether such head motion controlled vehicles affect performance and whether they are comfortable for users as these techniques may elicit simulator sickness. Therefore, in the present study, I simulated head motion

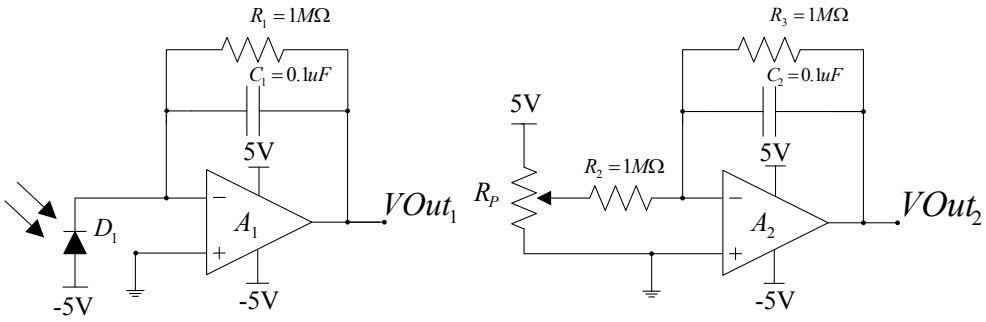


Figure 6-1: Circuit Diagrams (Left: Photodiode Circuit, Right: Potentiometer Circuit)

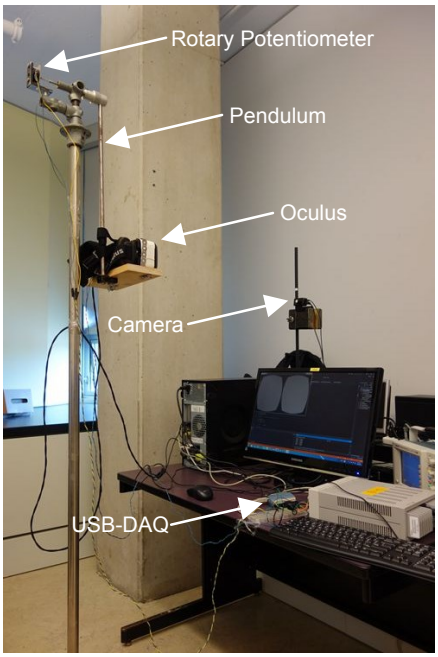


Figure 6-2: The Setup of the Experiment

controlled quadcopter scenarios using HMDs in a virtual environment. The VR paradigm enabled me to experimentally control the degree of latency and systematically assess the effects of latency in head motion controlled quadcopters.

The VR paradigm offers several advantages compared to experiments using real quadcopters. Firstly, latency can be easily controlled by setting appropriate gains in the simulation equations of quadcopters. Secondly, I avoid losing or damaging a drone when users are unable to control the quadcopter due to technical failures or potential simulator sickness caused by experiments. Third, complex testing environments can be easily set up and flight data can be conveniently and accurately logged.

The goal of the present research is (1) to estimate the end-to-end latency in the Oculus Rift DK2, verify the results of latency estimation done by (Raaen and Kjellmo, 2015; Kijima and Miyajima, 2016) and determine if the motion-to-photon latency plays dominant role in eliciting simulator sickness; (2) to assess users' flight performance and the degree of simulator sickness given various levels of the visual and control latency introduced by the aerial dynamics of simulated quadcopters. I also aimed to investigate whether people feel comfortable teleoperating a quadcopter using an HMD and whether they can adapt to the latency with practice. The results of the research may serve as guidelines on the design of head motion controlled vehicles.

6.2 Method

6.2.1 Estimating the Motion-to-Photon Latency

The Oculus Rift DK2 was mounted on the swing arm of a pendulum to introduce a damped sinusoidal motion to the HMD. The luminance of the display presented on the HMD was modulated either by the roll angle or the lateral translation of the HMD for rotational la-

tency estimation and translational latency estimation, respectively. I used a rotary potentiometer to capture the physical motion of the pendulum, and a photodiode was used to measure the changes of the intensity of the gradient stimulus rendered on the HMD during pendulum swing. The motion-to-photon latency was estimated by using the Fast Fourier Transform (FFT) algorithm to estimate the phase shift between these two signals at the frequency of the pendulum swing.

I present the circuit diagrams for the experiment in Figure 6-1. The current from the photodiode D_1 (Osram SFH206K) was amplified by an operational amplifier A_1 (Texas Instrument LF411-N) in transimpedance mode. The photodiode was negatively biased (-5 V). Since the amplification circuit introduced phase lag to the photodiode signal, I introduced the same phase lag to the signal of the potentiometer R_P (Novotechnik P4500) by implementing an active first order low-pass filter with a second LF411-N A_2 operating in inverting mode. I verified the phase lag of both circuits in the circuit simulation software MPLAB Mindi and measured the time delay of the physical circuits by giving both circuits a sinusoidal input of 0.7 Hz and verifying the phase lag.

I rendered the stimulus and capturing signals with a custom C++ application in Visual Studio 2013 with DirectX 11, the Oculus 0.5.5 SDK and the Universal Library of the USB-DAQ module for the Micro Computing USB-1208LS. For rotational latency estimation, I mapped the roll angle Φ of the Oculus Rift DK2 to the intensity I of the gray-scale image rendered on the display using the equation:

$$I = (\Phi + 90°)/180° \tag{6.1}$$

with $0 \leq I \leq 1.0$ and $-90° \leq \Phi \leq 90°$.

For translational latency estimation, I first mapped the ratio between the lateral translation x of the Oculus Rift DK2 and the length l of the swing arm to the roll angle using the asind() function; then, the calculated roll angle was mapped to the intensity I of the grayscale image on the display:

$$I = (asind(x/l) + 90°)/180° \tag{6.2}$$

with $-0.49\,m \leq X \leq 0.49\,m$ and $L = 0.49\,m$.

The mapped intensity was presented with the function ClearRenderTargetView() in DirectX 11. Thus, the stimulus presented on the display was essentially a motion-dependent gradient stimulus and the mapping was done for each rendering cycle. The custom application was hosted on a computer running Windows 7 with an Intel i7 2.8 GHz CPU, 4 GB memory and an AMD Radeon HD 6850 graphics card.

The captured periodic damped sinusoidal signals of photodiode and rotary potentiometer were analyzed using the following method. I first estimated the period of the pendulum swing T for each measurement by applying auto-correlation on the signal of the rotary potentiometer. I then performed the FFT transform on both signals of the photodiode and the rotary potentiometer. In the frequency domain, I determined that the bin with the maximum magnitude is the frequency component that corresponds to the frequency of the pendulum swing (Di Luca, 2010), which is approximately 0.7 Hz. Finally, I calculated the phases of the frequency components of both signals that correspond to the pendulum swing frequency.

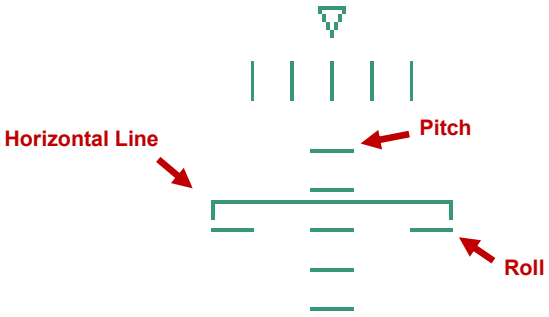


Figure 6-3: HUD for the Quadcopter

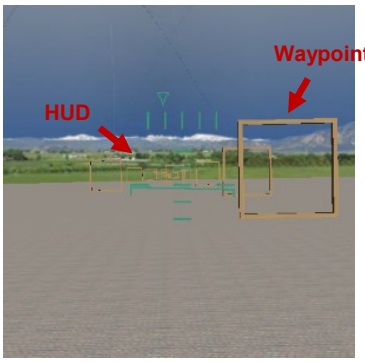


Figure 6-4: Testing Environment

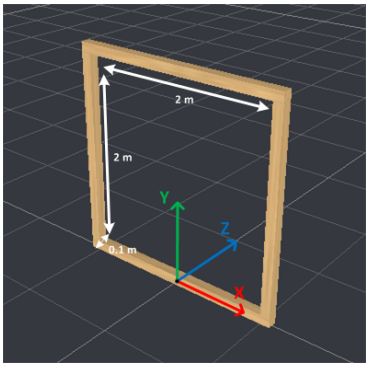


Figure 6-5: Waypoint and Its Coordinate

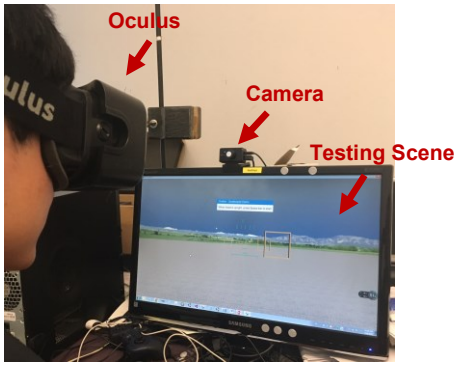


Figure 6-6: Experiment Setup

Table 6-1: Latency Levels

Latency Level	1	2	3	4	5
Gains k_Θ, k_Φ	32.5	15.6	10.5	7.9	6.5
Latency (s)	0.2	0.4	0.6	0.8	1.0

Table 6-2: Experimental Session Order

Participant	Day 1	Day 2	Day 3	Day 4	Day 5
P1	Training, Latency 3	Latency 5	Latency 1	Latency 2	Latency 4
P2	Training, Latency 1	Latency 4	Latency 5	Latency 3	Latency 2
P3	Training, Latency 1	Latency 2	Latency 5	Latency 3	Latency 4
P4	Training, Latency 5	Latency 3	Latency 2	Latency 1	Latency 4
P5	Training, Latency 5	Latency 2	Latency 1	Latency 3	Latency 4
P6	Training, Latency 5	Latency 4	Latency 1	Latency 3	Latency 2
P7	Training, Latency 2	Latency 3	Latency 4	Latency 1	Latency 5
P8	Training, Latency 3	Latency 4	Latency 5	Latency 2	Latency 1
P9	Training, Latency 4	Latency 2	Latency 3	Latency 1	Latency 5

The latency L was obtained by subtracting the phases and the time delay was calculated by dividing the phase shift by 2π and multiplying the result with the period of the potentiometer signal:

$$L = (\theta_d - \theta_p)/2\pi * T \tag{6.3}$$

where θ_d and θ_p are the phases (in radians) of the signals of the photodiode and the rotary potentiometer respectively.

6.2.2 Simulating a Head Motion Controlled Quadcopter

A. Software

The visual and control latency for head motion controlled quadcopters is defined as the time interval between when a desired tilt angle of the quadcopter is given by the pitch and roll of a user's head and when the tilt angle of the quadcopter reaches the desired tilt angle. To isolate how the visual and control latency elicits simulator sickness and affects flight performance, I focused on the simulation of a quadcopter that only allowed tilt (pitch and roll) and translation while yaw and altitude control were disabled. The application for simulating a head motion controlled quadcopter was developed using Python 2.7 based on the helicopter transport method in the Worldviz Vizard 5.0.

The position of a quadcopter flying at a fixed height can be represented by $\boldsymbol{P} = (x, z)$. The attitude can be represented by $\boldsymbol{\Theta} = (\Theta, \Phi)$, where Θ and Φ are the pitch and the roll of the quadcopter. The control input to a quadcopter can be defined as $\boldsymbol{u} = (\Theta_D, \Phi_D)$, which are the desired pitch and the desired roll of the quadcopter given by the

roll and the pitch of a user's head tracked by an HMD. To translate, a quadcopter first tilts itself. The tilt of the quadcopter at a fixed height can be simulated by the following set of equations:

$$\begin{cases} \dot{\Theta} = k_{\Theta}(\Theta_D - \Theta) \\ \dot{\Phi} = k_{\Phi}(\Phi_D - \Phi) \end{cases} \tag{6.4}$$

The angular rates of the pitch and the roll ($\dot{\Theta}$ and $\dot{\Phi}$) are proportional to the differences between the setpoints (pitch setpoint Θ_D and roll setpoint Φ_D) and their actual instantaneous values (pitch Θ and roll Φ). The constants of proportionality are the respective gains k_{Θ} and k_{Φ}. These gains control how fast the actual pitch Θ and roll Φ converge to their setpoints Θ_D and Φ_D.

The simple dynamics equations allowed me to easily set the latency for simulation by modifying the gains k_{Θ} and k_{Φ}. I measured the latency as the rise time of the step responses of the tiled angles (Θ and Φ) from 0˚ to 10˚ given the corresponding step inputs from 0˚ to 10˚. In Table 6-1, I present the five different latency levels used in the experiment and their corresponding gains. The latency values for the experiments were within the normal range of the step response of attitude control of a quadcopter. For example, the measured rise time of pitch and roll of a real quadcopter were approximately 0.2 s and 0.4 s respectively, when 10˚ step inputs were given during indoor flight (Kugelberg, 2016). The gains were determined based on a rate of 75 Hz for updating the equations. However, it should be noted that the latency with a given gain is dynamic. For example, if a user tilts head for 1˚, the duration for the quadcopter to adjust its attitude to reach the setpoint is much smaller than if the user tilts head for 10˚. Thus, the latency is also dependent on users'

behavior to tilt their heads and the gains only set the upper bounds for the latency values. When a quadcopter is tilted, it starts to translate. The translation motion is simulated by:

$$\begin{cases} \ddot{x}_B = (\Phi / \Phi_{max})\, a - \dot{x}_B d \\ \ddot{z}_B = (\Theta / \Theta_{max})\, a - \dot{z}_B d \end{cases} \tag{6.5}$$

The first terms in the equations of $\ddot{x}_B$ and $\ddot{z}_B$ show that the translation accelerations of the quadcopter are proportional to the tilt angles of the quadcopter scaled by a ($a = 10\ m/s^2$), which are the maximum translational accelerations of the quadcopter. Drag terms $\dot{x}_B d$ and $\dot{z}_B d$ – proportional to the velocities ($\dot{x}_B$ and $\dot{z}_B$) scaled by a factor d ($d = 0.05$) – were introduced to give inertia to the acceleration of the quadcopter. The maximum tilt angles Φ_{max} and Θ_{max} of a quadcopter are limited ($\Phi_{max} = 10^{\circ}$ and $\Theta_{max} = 10^{\circ}$). During real-time simulation, the accelerations and the angular rates were integrated over time at the rate of 75 Hz and consequently the quadcopter moved in the virtual environment.

During my initial trials, I found that one difficulty operating the quadcopter from a first-person view was to judge the attitude of the quadcopter. People had difficulty in hovering the quadcopter and making precise maneuvers. To address this issue, I designed a simple Heads-Up Display (HUD), as shown in Figure 6-3, which indicates the attitude of the quadcopter. The horizontal line represents the pitch by moving up and down and the roll by tilting.

I designed an experiment scene (Figure 6-4) that contained a stone-textured ground and a skydome. Similar to the experiment environment by Pittman and LaViola (2014), which used archways, I placed 100 square waypoints (Figure 6-5) in the scene at a height of 5 m and the altitude of the quadcopter was fixed at 6 m. The inner dimension of each

waypoint was 2 m (W) × 2 m (H) × 0.1 m (D). The longitudinal distances (*z*-axis) between these waypoints were 5 m. The lateral positions (*x*-axis) were randomized in the interval of ± 5 m.

B. Hardware

I used the Oculus Rift DK2 to track the users' head motion and present the simulated drone control scenario. This HMD uses a hybrid optical-inertial tracker (LaValle *et al*., 2014), which consists of an inertial measurement unit and a camera with an infrared lens, to track users' head motion at a sampling rate of approximately 75 Hz. The simulation was hosted on a computer running Windows 7 with an Intel i7 2.8 GHz CPU, 4 GB memory and an AMD Radeon HD 6850 graphics card. Figure 6-6 presents the experiment setup.

6.3 Experiment 6-1: Estimating the Motion-to-Photon Latency in HMDs

6.3.1 Introduction

The goal of the experiment is to estimate the motion-to-photon latency in the Oculus Rift DK2.

6.3.2 Procedure

Figure 6-2 shows the setup-up of the experiment. The shafts of the rotary potentiometer and the pendulum were coupled to measure the motion of pendulum swing. The Oculus Rift DK2 was mounted on the swing arm of the pendulum. I took out the left lens of the Oculus Rift DK2 and the photodiode was mounted into the opening with approximately

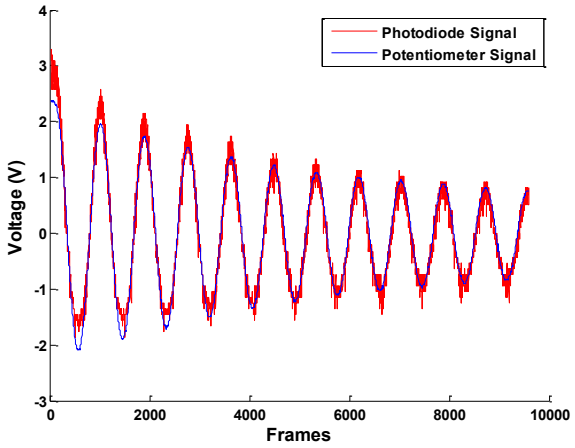


Figure 6-7: Exemplar Data Traces

Table 6-3: Results of Estimated Latencies (DP - Dynamic Prediction and TW - Time Warping)

Angle	DP	TW	Translation (ms)	Rotation (ms)
60°	OFF	OFF	2.7 ± 0.3	1.0 ± 0.4
	ON	OFF	2.7 ± 0.8	1.0 ± 0.5
	OFF	ON	3.1 ± 0.3	0.9 ± 0.3
	ON	ON	4.0 ± 0.2	1.3 ± 0.7
75°	OFF	OFF	4.4 ± 0.5	4.3 ± 0.4
	ON	OFF	5.0 ± 1.3	5.2 ± 1.4
	OFF	ON	5.3 ± 0.4	5.0 ± 1.4
	ON	ON	5.3 ± 0.4	4.8 ± 1.0
90°	OFF	OFF	6.5 ± 1.1	9.7 ± 0.7
	ON	OFF	6.3 ± 0.8	10.3 ± 0.9
	OFF	ON	5.6 ± 1.2	9.6 ± 0.8
	ON	ON	7.2 ± 0.5	9.7 ± 1.5

0.5 cm to the display panel. The signals from the photodiode and the rotary potentiometer were wired to two analog measurement channels of the data acquisition module. The distance from the front surface of the mounted Oculus Rift DK2 to the lens of its tracking

camera was 76 cm. The tracking camera was enabled throughout the experiment. To perform one measurement, I first pressed a key on the keyboard to initiate 16 s of data collection at 600 samples per second per channel and immediately dropped the pendulum afterwards. The signal capture automatically stopped when the 16 s duration elapsed. I repeated the measurement procedure for all combinations of (a) different initial roll angles of 60°, 75° and 90°, (b) Dynamic Prediction on or off and (c) Time Warping on or off. The latter two features were designed to reduce latency for predictable signals. Both the translational latency and the rotational latency were estimated in separate trials. Thus, in total, there were twenty-four conditions. For each condition, I repeated the measurements eight times. The method described in Section 6.2.1 was applied on captured signals in Matlab 2014a to estimate the motion-to-photon latency in each condition.

6.3.3 Results

Figure 6-7 presents exemplar data traces of the captured photodiode and potentiometer signals. These two signals were normalized to show that the phase lag between the photodiode signal and potentiometer signal is minimal. Thus, it is nearly impossible to manually determine the latency by examining the indices of the data components and the FFT analysis is needed. Table 6-3 summarizes the latency estimates from the FFT analysis. I can see that latencies are proportional to the roll angles from which the pendulum was dropped. Thus, performing sudden and rapid head movements may elicit stronger disorientation to users in VR environments than slower head movements do. Given different roll angles, the rotation latency seems to have a steeper slope than translation latency does. In all cases,

turning on both Dynamic Prediction and Time Warping did not reduce the latency. A major difference between the present study compared with those by (Raaen and Kjellmo, 2015; Kijima and Miyajima, 2016) is that I did not use a virtual scene for estimating latency. I cleared each frame with the function ClearRenderTargetView(). Thus, there were no vertex shader and pixel shader involved. This may be one reason why Dynamic Prediction and Time Warping did not reduce the latency.

6.4 Experiment 6-2: Effects of Visual and Control Latency on Piloting Quadcopters using HMDs

6.4.1 Introduction

The goal of my experiment was to investigate the flight performance of the participants and the degree of simulator sickness as a function of various levels of latency. I hypothesized that higher latency would degrade flight performance and elicit more simulator sickness. I reasoned that flight performance may improve and simulator sickness may relieve across sessions.

I adopted a repeated-measures design for the experiment. The independent factor for experiment was the latency of dynamics controlled by setting the gains k_Θ and k_Φ in the simulation equations of the quadcopter in (6.5), which resulted in different latency values (Table 6-1). To balance the order effect of treatments, I randomized the order of latency levels for each participant as shown in Table 6-2.

6.4.2 Participants

Ten people participated in the experiment (age: 20 - 38). All had normal or corrected-to-normal vision and were naïve to the goals of the experiment. Informed consent was obtained from all participants in accordance with a protocol approved by the Human Participants Review Subcommittee at the University.

6.4.3 Procedure

The experiments were conducted on five different days. On each experiment day, a participant completed one experiment session. The five experiment sessions for each participant were completed within two weeks. Simulator sickness questionnaires (SSQs) (Kennedy *et al*., 1993) were completed before and after each session.

During the experiment sessions, the participant wore the Oculus Rift DK2 and sat approximately 60 cm in front of the monitor, on which the camera of the HMD was attached. A researcher sat beside the participant and observed the virtual scene the participant saw on the computer monitor. At the beginning of an experiment session, a calibration procedure was conducted using the HUD (Figure 6-3). Participants were asked to level their head by observing the horizontal line that indicated the pitch and the roll of their heads on the HUD such that the horizontal line overlapped with the three short horizontal lines. This indicated that the yaw and the pitch of the participant's head was close to zero. When calibration was completed, the researcher pressed the start button to initiate the flight and the data recording started immediately. The goal for participants was to pass through the centers of all waypoints and they were only allowed to move the quadcopter by pitch and

roll movements. In case that they missed a waypoint, they were advised not to backtrack. Participants were required to enter a pink bounding volume when they reached the last waypoint. Upon entering the pink bounding volume, the data recording stopped immediately, and the task was considered as completed. On the first day (as shown in Table 6-2), a training session was conducted to teach participants the operation of the quadcopter using the HMD. The training scene was a simplified version of the actual experiment scene and only consisted of one waypoint and a pink bounding volume. The latency value was set to zero by directly mapping the pitch setpoint Θ_D and the roll setpoint Φ_D given by users' tracked head orientation to the pitch Θ and the roll Φ of the quadcopter, respectively. Pre-experiment SSQs were completed after the training session for the first day. The experiment application collected the timestamps, the positions and the Euler angles of the quadcopter for data analysis.

6.4.4 Metrics

The data collected from the experiment enabled me to extract a wide range of parameters to assess the degree of simulator sickness and flight performance of participants. Specifically, I used the following parameters as the measures:

- **SSQ Scores:**

  The standard SSQ was used as the subjective measure to evaluate the degree of simulator sickness elicited by the experiment.

- **Task completion time $\boldsymbol{T}$ $(\boldsymbol{s})$:**

  The duration from the start of the flight to the end of the flight.

- **Average flight speed S (m/s):**

The average flight speed computed by dividing the total length of the flight path by task completion time T.

- **Smoothness of the flight path D (m):**

The mean value of the lateral distance (x-axis) of the actual flight path to the optimal flight path. I defined the optimal flight path as the shortest distance between the centers of two adjacent waypoints.

- **Number of waypoints passed N_w:**

The number of waypoints that participants successfully passed. Since I had 9 participants, the ideal value of the total number S_w of the waypoints passed by all participants was nine hundred.

- **Number of collisions N_c:**

The number of the collisions with the frames of the waypoints. I assumed the quadcopter had a dimension of 0.3 m (W) × 0.1 m (H) × 0.3 m (D), which is a typical size of a civilian quadcopter. Participants were required to pass through the centers of waypoints as they were unable to judge whether the quadcopter would collide with the waypoints from the first-person view. The collisions were determined by computing the intersection of the bounding boxes of the quadcopter and the frames of the waypoints. The total number of collisions by all participants is denoted as S_c.

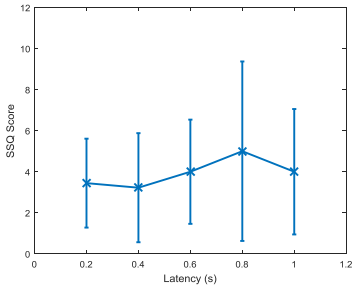


Figure 6-8: SSQ Score Increase by Latency

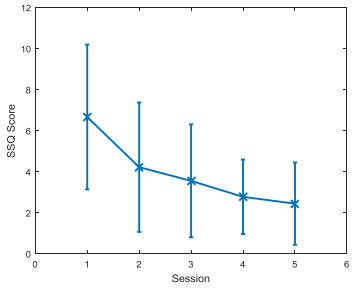


Figure 6-9: SSQ Score Increase by Session

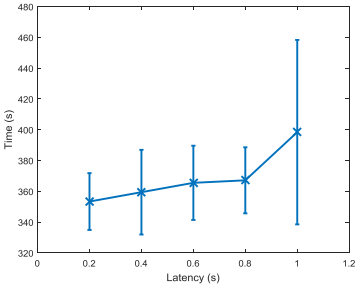


Figure 6-10: Task Completion Time by Latency

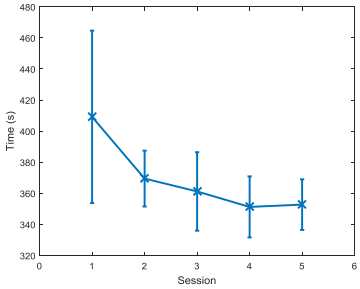


Figure 6-11: Task Completion Time by Session

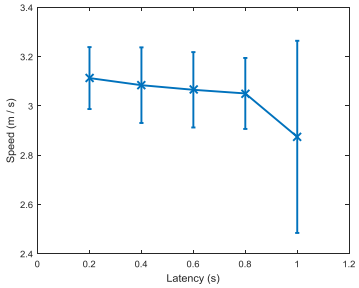


Figure 6-12: Average Flight Speed by Latency

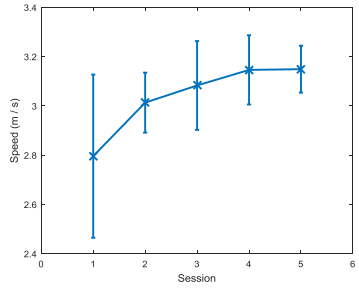


Figure 6-13: Average Flight Speed by Session

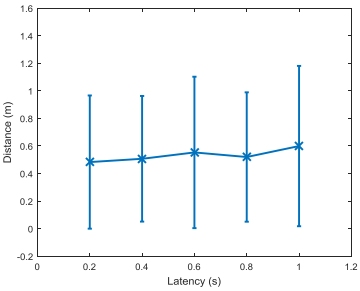


Figure 6-14: Path Smoothness by Latency

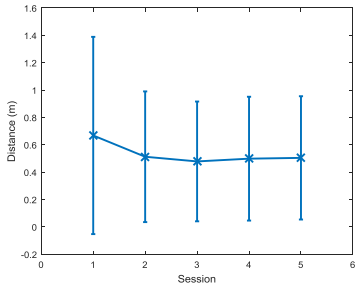


Figure 6-15: Path Smoothness by Session

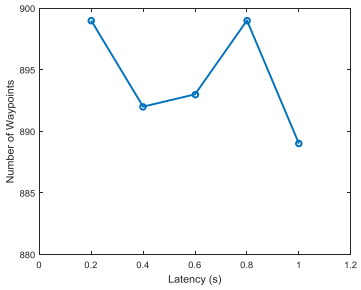


Figure 6-16: Total Number of Waypoints Passed by Latency

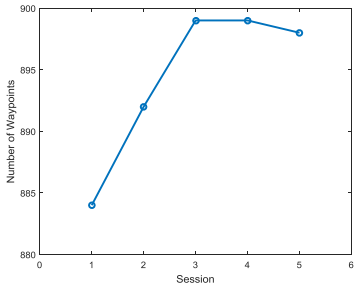


Figure 6-17: Total Number of Waypoints Passed by Session

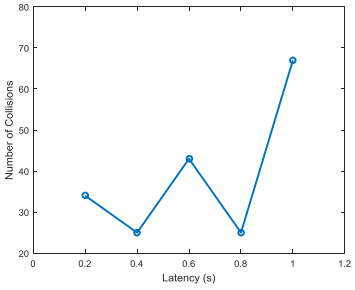


Figure 6-18: Total Number of Collisions by Latency

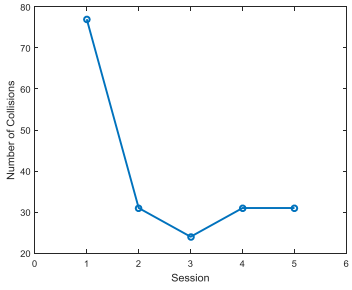


Figure 6-19: Total Number of Collisions by Session

Table 6-4: Results of Three-way ANOVA Analyses (significant p-values are in bold and shaded)

		SSQ	$T\ (s)$	$S\ (m/s)$	$D\ (m)$	N_w	N_c
Participant	$F(8,28)$	8.93	3.60	3.47	16.29	2.41	2.54
	p	**<0.001**	**0.005**	**0.007**	**<0.001**	**0.04**	**0.03**
	η_p^2	0.719	0.507	0.498	0.823	0.408	0.421
Session	$F(4,28)$	8.85	6.96	6.33	6.19	1.25	1.7
	p	**<0.001**	**<0.001**	**<0.001**	**0.001**	0.31	0.18
	η_p^2	0.559	0.498	0.475	0.469	0.151	0.196
Latency	$F(4,28)$	2.09	3.12	2.04	1.56	0.49	0.97
	p	0.11	**0.03**	0.12	0.21	0.74	0.44
	η_p^2	0.230	0.309	0.226	0.182	0.065	0.122

6.4.5 Results

One person withdrew from the experiment after attending the training session due to simulator sickness. Thus, I had valid data from nine participants (age: 20 - 38, 7 males, 2 females).

I grouped the extracted parameters by latency levels and by the order of experiment sessions to show whether the extracted parameters are more related to latency levels or session orders (error bars in Figure 6-8 - Figure 6-15) denote standard deviation.

Since the SSQ scores either remained the same or increased after the completion of an experiment session, I took absolute values of the differences between pre- and post-experiment SSQ scores for all sessions to remove pre-experiment biases. In general, SSQ scores were high when latency levels were high (Figure 6-8), which showed that high latency increased simulator sickness. Because of the continuous mapping between the head orientation and the tilt of the quadcopter, the degree of simulator sickness was subtle for all five latency conditions. In terms of sessions, I found that people adapted across experiment sessions and the SSQ scores gradually declined as more experiment sessions were completed by participants (Figure 6-9).

High latency values also increased task completion time T (s) (Figure 6-10) and reduced average flight speed S (m/s) (Figure 6-12). There were two reasons. First, when latency was high, the tilt motion of the quadcopter is slow. Hence, the acceleration of the quadcopter was also slow as the acceleration is dependent on the tilt of the quadcopter. Second, when latency was high, participants could have much difficulty in maneuvering

the quadcopter. Participants may have to perform more rolls to point the quadcopter to the center of a waypoint before flying it through. Due to learning and adaptation effects, task completion time gradually shortened (Figure 6-11) and average flight speed also increased (Figure 6-13) as participants attended more experiment sessions.

Different latency conditions slightly affected the smoothness of the flight path $D\ (m)$ (Figure 6-14). High latency generally led to less smooth flight paths. I also found that as more experiment sessions were conducted, flight path also became smoother (Figure 6-15). This indicated that participants performed fewer rolls, which also can be attributed to learning and adaptation effects.

The total number of waypoints passed S_w and the total number of collisions S_c were coarse parameters compared with previous four parameters. These two parameters were only affected when participants deviated too much from the centers for the waypoints, which resulted in missing or colliding with the waypoints. Thus, I did not observe any meaningful trends when grouping these two parameters by latency (Figure 6-16 and Figure 6-18). But both these parameters improved in terms of sessions (Figure 6-17 and Figure 6-19), which indicated that participants became more skilled in controlling quadcopters with more practice.

In general, when latency was higher, the standard deviation of an extracted parameter was usually large. This showed that the tolerance of high latency differed between participants. Similarly, I also observed that the standard deviations of the extracted parameters of the first sessions were high. This indicated that some people were initially good at piloting

a quadcopter using an HMD while others were not. Standard deviations decreased in later sessions, which indicated the performance between participants became similar.

In Table 6-4, I summarized the results of the Three-way ANOVA analyses using the function anovan() in Matlab to study the effects of the extracted parameters. The model included Session and Latency as fixed factors while Participant was treated as a random factor. This is similar to the model presented by Blissing *et al*. (2016). Effect sizes were reported using partial eta squared η_p^2. In general, Participant had significant effects on all parameters, which demonstrated that tolerance and skills to the quadcopter control scenario differed between participants. Session had significant effects on SSQ scores, task completion time $T\ (s)$, average flight speed $S\ (m/s)$ and smoothness of the flight path $D\ (m)$ due to learning and adaptation. Latency had a significant effect on task completion time $T\ (s)$. One reason that Latency did not influence other parameters was that participants consistently adapted to the quadcopter scenario as they attended more experiment sessions and the latency effects were therefore mitigated. Second, the visual and control latency is dynamic and dependent on how users tilt their heads. Moving heads in small angles with slow motion results in lower latency compared to making abrupt head movements in large angles.

Pairwise comparisons on fixed factors were conducted using Tukey's range tests. These tests showed that for the factor Session, there were significant differences between the first session and the rest four sessions on SSQ scores, task completion time $T\ (s)$ and smoothness of the flight path $D\ (m)$. A significant difference was also found on average flight speed $S\ (m/s)$ between the first session and the last three sessions. This may suggest

that participants made substantial progress in adapting to the quadcopter scenario after they completed the first session. In terms of the factor Latency, there was a significant difference between the latency condition of 0.2 s and the latency condition of 1 s on task completion time $T\ (s)$, which suggested that high latency increased task completion time $T\ (s)$.

6.5 Discussion

In this chapter, I first presented a method for estimating the motion-to-photon latency of HMDs. Similar to the results reported by Raaen and Kjellmo (2015) and Kijima and Miyajima (2016), my method also showed that the Oculus Rift DK2 has a very low latency, which could be the result of the prediction algorithms described by LaValle *et al*. (2014). While the present study focused on estimating the latency of the Oculus Rift DK2, the proposed method is general and can be applied to estimate the latency of other HMDs. A limitation of the proposed method is that the frequency of the pendulum is fixed. To test whether the latency of the Oculus Rift DK2 is frequency-dependent, a motorized platform is needed (Di Luca, 2010). In addition, the accuracy of the proposed estimation approach needs to be addressed in future research. This study also showed that the motion-to-photon latency is only responsible for a small portion of the overall latency in head motion controlled quadcopter scenarios and visual and control latency plays a dominant role in such cases.

I then presented a VR paradigm to systematically evaluate the effects of the visual and control latency introduced by dynamics of quadcopter. I showed that a latency value with an upper bound of 1 s only elicits subtle simulator sickness when head motion was

continuously mapped to the motion of the quadcopter. In addition, high latency values resulted in worse flight performance and higher level of simulator sickness. I also showed as participants attended more experiment sessions, they became more tolerant to the head motion controlled drone scenarios. Lower SSQ scores were reported and flight performance also improved. These results have verified my hypotheses that higher latency would degrade flight performance and elicit more simulator sickness and that tolerance to the quadcopter scenario and flight performance may differ between participants.

The present study suggests piloting a quadcopter using an HMD is feasible in terms of tolerance to visual and control latency, but training is needed for people to efficiently and comfortably operate the quadcopter with the interface. Selecting people inherently good at the quadcopter control scenario as pilots facilitates the training process. In addition, using a quadcopter that has a fast response to changing tilt commands would both improve the flight performance and reduce simulator sickness. Additional recommendations in designing such systems are to use the HMDs with low motion-to-photon latency and to continuously map head motion to quadcopter motion.

Watanabe and Takahashi (2018) presented a user study that assessed simulator sickness using videos presented on an HMD that simulated a hovering quadcopter disturbed by winds. Further research should study flight performance and simulator sickness when a quadcopter is subject to air disturbance when users perform flight maneuvers using VR. In addition, the influence of the yaw and the altitude control of a quadcopter on flight performance and simulator sickness also needs to be studied. In the current study, the scene setup

was relatively simple, so it also will be interesting to investigate the relationship between scene complexity and simulator sickness. Finally, an experiment with real quadcopters also needs to be conducted.

Chapter 7
Conclusion and Future Work

The overall aim of the dissertation was to contribute to the improvement of the naturalness and the intuitiveness of the interaction between humans and virtual environments through the design of new interfaces for VR and performing related user studies.

To realize the aim, this dissertation presented two interaction techniques in VR. The first one is a locomotion technique interacting with a large-scale projective display, known as the WISE while the second technique is a head gesture recognition approach on HMDs to interact with VR. I evaluated these two approaches through novel user studies. For the locomotion technique, I designed a target pursuit task that had participants to pursue a rolling ball in a virtual scene. The head gesture recognition technique was first evaluated in terms of its offline classification performance and the latency to recognize head gestures. Then, its usability and user preference were compared with a dynamic hand gesture recognition approach proposed in the dissertation as well as a conventional gamepad approach through a Yes/No task in virtual environments.

This dissertation also presented two psychophysical studies concerning the role of stereopsis in avoiding virtual obstacles during walking in VR and the effects of visual and control latency in piloting a quadcopter using an HMD. These studies have revealed some interesting results. For example, I found that stereopsis helped participants to make more accurate movements when stepping over obstacles and gaps. I also found that high visual and control latency worsened people's flight performance and elicited simulator sickness,

but people were able to adapt to the control scenario through practice, which led to improved flight performance and lessened simulator sickness. As part of the latter project, I also proposed a method for estimating the motion-to-photon latency in HMDs. The method is general and can be used to estimate the motion-to-photon latency in other HMDs with minimal hardware design and programming efforts.

While previous studies on treadmill speed adaptation built their own walking speed estimation model (Yoon *et al*., 2012; Wiens *et al*., 2017) to adapt the speed of a treadmill, the work in this dissertation showed that it is possible to use a machine learning approach to classify mid-swing speed of foot motion to implement a locomotion interface for users to walk in VR. Although the current approach was implemented based on a linear treadmill, it also can be integrated to the control of multi-dimensional treadmills to potentially improve the control performance of such treadmills. To validate the extension, the performance of the proposed controller needs to be compared with existing controllers on the same hardware (treadmills and trackers) in future research.

Previous research on the role of stereopsis was primarily conducted on tasks related to hand-eye coordination (Fielder and Moseley, 1996) and on walking tasks conducted in limited physical space (Patla *et al*., 2002; Loomis *et al*., 2006; Hayhoe *et al*., 2009; Chapman *et al*., 2012). My research showed that stereopsis also helped people to step over virtual obstacles and gaps more accurately under constant motion during continuous walking in virtual environments. One implication of the results is that it reinforces the importance

of rendering stereoscopic images to users during continuous locomotion tasks in VR. Rendering stereoscopic images to both eyes requires additional rendering passes from two different eye positions, and my research showed it is beneficial to render stereoscopic images despite the additional computational expenses as stereoscopic images enable users to perform more accurate walking movements in virtual environments. For users themselves, walking under the stereoscopic viewing condition is more similar to real-life experience compared to walking under the non-stereoscopic viewing condition.

The head gesture interface presented in the dissertation was the first head gesture recognition algorithm that has been implemented on HMDs using the Cascaded Hidden Markov Models to interact with VR. A major contribution of this work is that it offers VR researchers and designers with an additional choice of VR interface. Since nearly all VR systems are head-tracked and head angular velocity data are readily available, this head gesture interface can be easily integrated into existing VR systems as it only requires head angular velocity data to work. As the head gesture interface was implemented in a fully-pipelined structure, it also can be easily implemented on hardware platforms on FPGAs and ASICs. To compare the head gesture interface, I also proposed a dynamic hand gesture recognition algorithm based on a static hand gesture recognition algorithm (Marin *et al*., 2016). Although my result showed that the hand gesture interface was not preferred compared to the head gesture interface and the gamepad interface, I speculated that the rejection of the interface might be related to the choice of the hand gestures representing Yes and

No. For future research, it is necessary to compare different hand gestures and determine what hand gestures are more preferred by users.

I implemented a simulation of head motion controlled quadcopter and, through user studies, found that the visual and control latency in such control scenarios resulted in simulator sickness and degraded flight performance. I pointed out that the visual and control latency is a different factor in head motion controlled vehicles compared to the visual latency and the control latency related to driving cars (Blissing *et al*., 2016) and piloting helicopters (Jennings *et al*., 2004). In the two latter cases, the latency is related to the time interval between hand motion and the resulting visual update on displays while in the former case, the visual and control latency are both related to the coupling between users' head motion and visual update on displays. The results of the research serve as guidelines for researchers or engineers to design such head motion controlled vehicles - it is important to select a quadcopter that is fast in response to control commands; and users are able to adapt to the control scenarios with practice and training.

To conclude the dissertation, I proposed the following future research topics:

- **Locomotion with the Wide Immersive Stereo Environment (WISE) that Supports Turning.** In Chapter 3, I presented a locomotion technique using the WISE and a linear treadmill based on a machine learning method. A limitation with the approach is that the it does not support turning during walking. To make the locomotion technique support turning, one can implement the sidestepping technique

or the head rotation technique developed by Vijayakar and Hollerbach (2002). Another approach to implement a turning strategy with the WISE using a turntable is currently underway. The basic idea of the approach is to use a turntable to re-orient a user to the center of the WISE when users make turns, which is similar to the idea of Redirected Walking in Place (Razzaque et al., 2002). In the technique of Redirected Walking in Place, a user is re-oriented to the center of a CAVE by manipulating the image contents presented in the CAVE to guide a user to re-orient while the turntable approach physically re-orient a user to the center of the WISE by monitoring their facing direction and mechanically re-orient a user. Walking on the turntable can be implemented based on existing Walking-in-Place (WIP) techniques (Templeman *et al*., 1999; Yan *et al*., 2004; Feasel *et al*., 2008; Wendt *et al*., 2010; Bruno *et al*., 2013; Wilson *et al*., 2014; Tregillus and Folmer, 2016; Bruno *et al*., 2017).

- **Avoiding Virtual Obstacles with Obstructed Visual Field.** In Chapter 4, I presented two VR experiments to investigate the role of stereopsis in avoiding virtual obstacles while walking. The experiment setup in the study can be further extended to study human walking behavior in avoiding virtual obstacles when visual fields are obstructed. The extension of the experiment setup is relatively simple - graphics shaders can be programmed to mask out the exterior of images presented on the WISE, resulting in field of views of various sizes. This will allow us to verify the results by Matthis and Fajen (2014) that walkers relied on visibility of the ground

at least two steps ahead to locomote normally. While their experiments were conducted using color blobs projected onto floors for participants to avoid, a VR experiment conducted using the WISE enables to investigate human walking behavior when avoiding volumetric objects, which are more similar to obstacles people encounter in their daily lives than planar objects.

- **Utility of the Head Gesture Interface and the Hand Gesture Interface.** In Chapter 5, I presented a head gesture interface and a hand gesture interface. A user study was conducted to demonstrate their utility in answering Yes/No questions in virtual environments. But these two interfaces may have broader utility that is worth further investigation. For example, one research direction is to investigate user preference to interact with virtual avatars using these two interfaces. A speech recognition interface also can be developed and compared with these two interfaces or a multimodal interface that integrates the head gesture interface, the hand gesture interface and the speech interface can be developed. In addition, these interfaces may serve as input methods to virtual environments to complete certain tasks, including browsing and giving confirmation. Finally, a hardware implementation of the head gesture interface based on FPGAs or ASICs can be developed to allow hardware centric head gesture recognition.
- **The Effects of Visual and Control Latency on a Real Quadcopter Piloted by an HMD.** In Chapter 6, I studied the effects of visual and control latency on piloting a quadcopter using an HMD in simulation. It is still necessary to study this topic on

real quadcopters and compare the effects with that of simulation. There are a few challenges to perform an experiment on real quadcopters. Firstly, the experiment needs a real quadcopter that can be programmed with different controllers that result in different latency values to set up experiment conditions. Or one can select a few quadcopters that are inherently different in designs that give different latency values. The concerns with the second approach are that the cost is high to buy multiple quadcopters than a single quadcopter and that more programming efforts are needed to adapt these quadcopters to support control and viewing by an HMD. Secondly, it is challenging to setup experiment scenes in the real world. Waypoints can be set up by balloons in the air, but due to factors such as the expense, the size of experiment site and weather conditions, the experiment setup in the real-world will be simpler than that in simulation. Thirdly, data logging is more difficult in the real-world. This needs to be done by GPS (Global Positioning System) modules embedded in quadcopters and waypoints (which are balloons) to record the positions of quadcopters and waypoints to allow further data analysis on the flight performance of users. Video cameras also may be used to monitor the flying path of quadcopters and positions of waypoints.

Bibliography

Abate, A.F., Acampora, G., Ricciardi, S., 2011. An interactive virtual guide for the AR based visit of archaeological sites. Journal of Visual Languages & Computing 22, 415–425.

Allison, R.S., Harris, L.R., Jenkin, M., Jasiobedzka, U., Zacher, J.E., 2001. Tolerance of temporal delay in virtual environments, in: Proceedings of IEEE Virtual Reality 2001. pp. 247–254.

Allison, R.S., Harris, L.R., Jenkin, M., Pintilie, G., Redlick, F., Zikovitz, D.C., 2000. First steps with a rideable computer, in: Proceedings of IEEE Virtual Reality 2000. pp. 169–175.

Allison, R.S., Zacher, J.E., Wang, D., Shu, J., 2004. Effects of network delay on a collaborative motor task with telehaptic and televisual feedback, in: Proceedings of VRCAI 2004. pp. 375–381.

Bentley, J.L., 1975. Multidimensional Binary Search Trees Used for Associative Searching. Commun. ACM 18, 509–517.

Blissing, B., Bruzelius, F., Eriksson, O., 2016. Effects of visual latency on vehicle driving behavior. ACM Trans. Appl. Percept. 14, 5:1–5:12.

Bruno, L., Pereira, J., Jorge, J., 2013. A new approach to walking in place, in: INTERACT 2013, Lecture Notes in Computer Science. pp. 370–387.

Bruno, L., Sousa, M., Ferreira, A., Pereira, J.M., Jorge, J., 2017. Hip-directed walking-in-place using a single depth camera. International Journal of Human-Computer Studies 105, 1–11.

Campbell, L.W., Becker, D.A., Azarbayejani, A., Bobick, A.F., Pentland, A., 1996. Invariant features for 3-D gesture recognition, in: Proceedings of the Second International Conference on Automatic Face and Gesture Recognition. pp. 157–162.

Chang, C., Lin, C., 2011. LIBSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol. 2, 27:1–27:27.

Chapman, G.J., Scally, A., Buckley, J.G., 2012. Importance of binocular vision in foot placement accuracy when stepping onto a floor-based target during gait initiation. Experimental Brain Research 216, 71–80.

Chen, C., Liang, J., Zhao, H., Hu, H., Tian, J., 2009. Factorial HMM and parallel HMM for gait recognition. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 39, 114–123.

Cheng, H., Yang, L., Liu, Z., 2016. Survey on 3D hand gesture recognition. IEEE Transactions on Circuits and Systems for Video Technology 26, 1659–1673.

Choi, I., Hawkes, E.W., Christensen, D.L., Ploch, C.J., Follmer, S., 2016. Wolverine: A wearable haptic interface for grasping in virtual reality, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 986–993.

Coomer, N., Bullard, S., Clinton, W., Williams, B., 2018. Evaluating the Effects of Four VR Locomotion Methods: Joystick, Arm-cycling, Point-tugging, and Teleporting, in: Proceedings of the 15th ACM Symposium on Applied Perception. pp. 7:1–7:8.

Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., 1993. 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.

de Vries, S.C., Padmos, P., 1997. Remotely controlled flying aided by a Head-slaved camera and HMD, TNO-report TM-97-B024. TNO Human Factors Research Institute.

Di Luca, M., 2010. New method to measure end-to-end delay of virtual reality. Presence: Teleoperators and Virtual Environments 19, 569–584.

Dodiya, J., Alexandrov, V.N., 2007. Perspectives on potential of sound in virtual environments, in: 2007 IEEE International Workshop on Haptic, Audio and Visual Environments and Games. pp. 15–20.

Doisy, G., Ronen, A., Edan, Y., 2017. Comparison of three different techniques for camera and motion control of a teleoperated robot. Applied Ergonomics 58, 527–534.

Feasel, J., Whitton, M.C., Wendt, J.D., 2008. LLCM-WIP: Low-Latency, Continuous-Motion Walking-in-Place, in: 2008 IEEE Symposium on 3D User Interfaces. pp. 97–104.

Fielder, A.R., Moseley, M.J., 1996. Does stereopsis matter in humans? Eye 10, 233–238.

Gamble, J.G., Rose, J., 1994. Human walking, 2nd ed. Baltimore : Williams & Wilkins.

Hartley, R., Zisserman, A., 2003. Multiple view geometry in computer vision, 2nd ed. Cambridge University Press.

Hayhoe, M., Gilliam, B., Chajka, K., Vecello, E., 2009. The role of binocular vision in walking. Visual neuroscience 26, 73–80.

Higuchi, K., Fujii, K., Rekimoto, J., 2013. Flying head: A head-synchronization mechanism for flying telepresence, in: 2013 23rd International Conference on Artificial Reality and Telexistence. pp. 28–34.

Hollerbach, J.M., 2002. Locomotion interfaces, in: Handbook of Virtual Environments: Design, Implementation, and Applications. Lawrence Erlbaum Associates, Inc., pp. 239–254.

Hossain, M., Jenkin, M., 2005. Recognizing hand-raising gestures using HMM, in: The 2nd Canadian Conference on Computer and Robot Vision. pp. 405–412.

Hsu, C., Lin, C., 2002. A comparison of methods for multiclass support vector machines. IEEE Transactions on Neural Networks 13, 415–425.

Iribe, B., 2013. Virtual reality – a new frontier in computing.

Iwata, H., 1999. Walking about virtual environments on an infinite floor, in: Proceedings IEEE Virtual Reality. pp. 286–293.

Iwata, H., Yano, H., Nakaizumi, F., 2001. Gait Master: a versatile locomotion interface for uneven virtual terrain, in: Proceedings IEEE of Virtual Reality 2001. pp. 131–137.

Jennings, S., Reid, L.D., Craig, G., Kruk, R.V., 2004. Time delays in visually coupled systems during flight test and simulation. Journal of Aircraft 41, 1327–1335.

Kalman, R.E., 1960. A new approach to linear filtering and prediction approach. Journal of Basic Engineering 82, 35–45.

Kang, H., 2013. Various approaches for driver and driving behavior monitoring: A review, in: 2013 IEEE International Conference on Computer Vision Workshops. pp. 616–623.

Kennedy, R.S., Lane, N.E., Berbaum, K.S., Lilienthal, M.G., 1993. Simulator Sickness Questionnaire: An enhanced method for quantifying simulator sickness. The International Journal of Aviation Psychology 3, 203–220.

Kijima, R., Miyajima, K., 2016. Measurement of Head Mounted Display's latency in rotation and side effect caused by lag compensation by simultaneous observation — An example result using Oculus Rift DK2, in: 2016 IEEE Virtual Reality. pp. 203–204.

Kitson, A., Hashemian, A.M., Stepanova, E.R., Kruijff, E., Riecke, B.E., 2017. Comparing leaning-based motion cueing interfaces for virtual reality locomotion, in: 2017 IEEE Symposium on 3D User Interfaces. pp. 73–82.

Kugelberg, I., 2016. Black-box modeling and attitude control of a quadcopter. Linköping University.

Lakens, D., 2013. Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and ANOVAs. Front. Psychol. 4. https://doi.org/10.3389/fpsyg.2013.00863

Langbehn, E., Lubos, P., Steinicke, F., 2018. Evaluation of locomotion techniques for room-Scale VR: Joystick, teleportation, and redirected walking, in: Virtual Reality International Conference.

LaValle, S.M., Yershova, A., Katsev, M., Antonov, M., 2014. Head tracking for the Oculus Rift, in: 2014 IEEE International Conference on Robotics and Automation. pp. 187–194.

LaViola, J.J., Jr., Feliz, D.A., Keefe, D.F., Zeleznik, R.C., 2001. Hands-free multi-scale navigation in virtual environments, in: Proceedings of the 2001 Symposium on Interactive 3D Graphics. pp. 9–15.

Lin, H., Venetsanopoulos, A.N., 1993. A weighted minimum distance classifier for pattern recognition, in: Proceedings of Canadian Conference on Electrical and Computer Engineering. pp. 904–907.

Liu, K., Chen, C., Jafari, R., Kehtarnavaz, N., 2014. Multi-HMM classification for hand gesture recognition using two differing modality sensors, in: 2014 IEEE Dallas Circuits and Systems Conference. pp. 1–4.

Loomis, J.M., Beall, A.C., Macuga, K.L., Kelly, J.W., Smith, R.S., 2006. Visual control of action without retinal optic flow. Psychological Science 17, 214–221.

Marin, G., Dominio, F., Zanuttigh, P., 2016. Hand gesture recognition with jointly calibrated Leap Motion and depth sensor. Multimed Tools Appl 75, 14991–15015.

Martins, H., Ventura, R., 2009. Immersive 3-D teleoperation of a search and rescue robot using a head-mounted display, in: 2009 IEEE Conference on Emerging Technologies Factory Automation. pp. 1–8.

Matsukura, H., Yoshida, H., Nakamoto, T., Ishida, H., 2010. Synchronized presentation of odor with airflow using olfactory display. Journal of Mechanical Science and Technology 24, 253–256.

Matthis, J.S., Fajen, B.R., 2014. Visual control of foot placement when walking over complex terrain. Journal of Experimental Psychology. Human Perception and Performance 40, 106–115.

McKee, S.P., Levi, D.M., Bowne, S.F., 1990. The imprecision of stereopsis. Vision Research, Optics Physiology and Vision 30, 1763–1779.

Medina, E., Fruland, R., Weghorst, S., 2008. Virtusphere: Walking in a human size VR "Hamster Ball." Proceedings of the Human Factors and Ergonomics Society 52nd Annual Meeting 52, 2102–2106.

Mollet, N., Chellali, R., 2008. Virtual and augmented reality with head-tracking for efficient teleoperation of groups of robots, in: 2008 International Conference on Cyberworlds. pp. 102–108.

Morimoto, C., Yacoob, Y., Davis, L., 1996. Recognition of head gestures using hidden Markov models, in: Proceedings of 13th International Conference on Pattern Recognition. pp. 461–465.

Morphew, M.E., Shively, J.R., Casey, D., 2004. Helmet-mounted displays for unmanned aerial vehicle control, in: Proc. SPIE 5442, Helmet- and Head-Mounted Displays IX: Technologies and Applications. pp. 93–103.

Nabiyouni, M., Saktheeswaran, A., Bowman, D.A., Karanth, A., 2015. Comparing the performance of natural, semi-natural, and non-natural locomotion techniques in virtual reality, in: 2015 IEEE Symposium on 3D User Interfaces. pp. 3–10.

Nilsson, N.C., Serafin, S., Laursen, M.H., Pedersen, K.S., Sikström, E., Nordahl, R., 2013. Tapping-In-Place: Increasing the naturalness of immersive walking-in-place locomotion through novel gestural input, in: 2013 IEEE Symposium on 3D User Interfaces. pp. 31–38.

Noma, H., Sugihara, T., Miyasato, T., 2000. Development of ground surface simulator for Tel-E-Merge system, in: Proceedings of IEEE Virtual Reality 2000. pp. 217–224.

Park, H.J., Lee, H.J., Kang, T.H., Moon, J.I., 2015. Study on automatic speed adaptation treadmills, in: 2015 15th International Conference on Control, Automation and Systems. pp. 1898–1900.

Park, J., Patel, A., Curtis, D., Teller, S., Ledlie, J., 2012. Online pose classification and walking speed estimation using handheld devices, in: Proceedings of the 2012 ACM Conference on Ubiquitous Computing. pp. 113–122.

Patla, A.E., Niechwiej, E., Racco, V., Goodale, M.A., 2002. Understanding the contribution of binocular vision to the control of adaptive locomotion. Experimental Brain Research 142, 551–561.

Pittman, C., LaViola, J.J., Jr., 2014. Exploring head tracked head mounted displays for first person robot teleoperation, in: Proceedings of the 19th International Conference on Intelligent User Interfaces. pp. 323–328.

Raaen, K., Kjellmo, I., 2015. Measuring latency in virtual reality systems, in: Entertainment Computing - ICEC 2015, Lecture Notes in Computer Science. pp. 457–462.

Rabiner, L.R., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77, 257–286.

Razzaque, S., Kohn, Z., Whitton, M.C., 2001. Redirected walking, in: Eurographics 2001 - Short Presentations.

Razzaque, S., Swapp, D., Slater, M., Whitton, M.C., Steed, A., 2002. Redirected walking in place, in: Proceedings of the Workshop on Virtual Environments 2002. pp. 123–130.

Robinett, W., Holloway, R., 1992. Implementation of flying, scaling and grabbing in virtual worlds, in: Proceedings of the 1992 Symposium on Interactive 3D Graphics. pp. 189–192.

Sarbolandi, H., Lefloch, D., Kolb, A., 2015. Kinect range sensing: Structured-light versus Time-of-Flight Kinect. Computer Vision and Image Understanding 139, 1–20.

Seo, M., Choi, S., Lee, S., Oh, E., Baek, J., Kang, S., 2017. Photosensor-based latency measurement system for head-mounted displays. Sensors 17, 1112.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A., 2011. Real-time human pose recognition in parts from single depth images, in: CVPR 2011. pp. 1297–1304.

Slater, M., Steed, A., Usoh, M., 1995. The virtual treadmill: a naturalistic metaphor for navigation in immersive virtual environments, in: Virtual Environments. pp. 135–148.

Smolyanskiy, N., Gonzalez-Franco, M., 2017. Stereoscopic first person view system for drone navigation. Frontiers in Robotics and AI 4.

Sokolova, M., Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Information Processing & Management 45, 427–437.

Souman, J.L., Robuffo Giordano, P., Frissen, I., De Luca, A., Ernst, M.O., 2010. Making virtual walking real: Perceptual evaluation of a new treadmill control algorithm. ACM Transactions on Applied Perception 7, 1–14.

Souman, J.L., Robuffo Giordano, P., Schwaiger, M., Frissen, I., Thümmel, T., Ulbrich, H., De Luca, A., Bülthoff, H.H., Ernst, M.O., 2008. CyberWalk: Enabling unconstrained omnidirectional walking through virtual environments. ACM Trans. Appl. Percept. 8, 25:1–25:22.

Steed, A., 2008. A simple method for estimating the latency of interactive, real-time graphics simulations, in: Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology. pp. 123–129.

Su, S.W., Wang, L., Celler, B.G., Savkin, A., 2005. Heart rate control during treadmill exercise, in: Proceedings of the 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. pp. 2471–2474.

Sutherland, I., 1968. A head-mounted three dimensional display, in: Proceedings of Fall Joint Computer Conference. pp. 757–764.

Taylor, R.M., II, Hudson, T.C., Seeger, A., Weber, H., Juliano, J., Helser, A.T., 2001. VRPN: A Device-independent, Network-transparent VR Peripheral System, in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology. pp. 55–61.

Teixeira, J.M., Ferreira, R., Santos, M., Teichrieb, V., 2014. Teleoperation using Google Glass and AR, Drone for structural inspection, in: 2014 XVI Symposium on Virtual and Augmented Reality. pp. 28–36.

Templeman, J.N., Denbrook, P.S., Sibert, L.E., 1999. Virtual locomotion: Walking in Place through virtual environments. Presence 8, 598–617.

Terven, J.R., Salas, J., Raducanu, B., 2014. Robust head gestures recognition for assistive technology, in: Pattern Recognition, Lecture Notes in Computer Science. pp. 152–161.

Tregillus, S., Folmer, E., 2016. VR-STEP: Walking-in-Place using inertial sensing for hands free navigation in mobile VR environments, in: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. pp. 1250–1255.

Vijayakar, A., Hollerbach, J.M., 2002. A proportional control strategy for realistic turning on linear treadmills, in: Proceedings 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. pp. 231–238.

von Zitzewitz, J., Bernhardt, M., Riener, R., 2007. A novel method for automatic treadmill speed adaptation. IEEE transactions on neural systems and rehabilitation engineering 15, 401–409.

Vu, D., Kövecses, J., Gosselin, C., 2017. Trajectory planning and control of a belt-driven locomotion interface for flat terrain walking and stair climbing, in: 2017 IEEE World Haptics Conference. pp. 189–194.

Ware, C., Arthur, K., Booth, K.S., 1993. Fish tank virtual reality, in: Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems. ACM, pp. 37–42.

Watanabe, K., Takahashi, M., 2018. Control System Design of a Quadrotor Suppressing the Virtual Reality Sickness, in: 2018 AIAA Modeling and Simulation Technologies Conference, AIAA SciTech Forum, (AIAA 2018-1916).

Wendt, J.D., Whitton, M.C., Brooks, F.P., 2010. GUD WIP: Gait-Understanding-Driven Walking-In-Place, in: 2010 IEEE Virtual Reality Conference. pp. 51–58.

Wiens, C., Denton, W., Schieber, M.N., Hartley, R., Marmelat, V., Myers, S.A., Yentes, J.M., 2017. Reliability of a feedback-controlled treadmill algorithm dependent on the user's behavior, in: 2017 IEEE International Conference on Electro Information Technology. pp. 545–550.

Williams, B., Bailey, S., Narasimham, G., Li, M., Bodenheimer, B., 2011. Evaluation of walking in place on a Wii balance board to explore a virtual environment. ACM Trans. Appl. Percept. 8, 19:1–19:14.

Wilson, P.T., Nguyen, K., Harris, A., Williams, B., 2014. Walking in Place Using the Microsoft Kinect to Explore a Large VE, in: Proceedings of the 13th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry. pp. 27–33.

Woodman, O.J., 2007. An introduction to inertial navigation, Technical Report 696. University of Cambridge.

Xu, D., 2006. A neural network approach for hand gesture recognition in virtual reality driving training system of SPG, in: 18th International Conference on Pattern Recognition. pp. 519–522.

Yan, L., Allison, R.S., Rushton, S.K., 2004. New simple virtual walking method - walking on the spot, in: 8th Annual Immersive Projection Technology Symposium Electronic Proceedings.

Yoon, J., Park, H., Damiano, D.L., 2012. A novel walking speed estimation scheme and its application to treadmill control for gait rehabilitation. Journal of Neuroengineering and Rehabilitation 9, 62.

Zielasko, D., Horn, S., Freitag, S., Weyers, B., Kuhlen, T.W., 2016. Evaluation of hands-free HMD-based navigation techniques for immersive data analysis, in: 2016 IEEE Symposium on 3D User Interfaces. pp. 113–119.