NTNU
Norwegian University of
Science and Technology

Redirected Walking, an Investigation into Noticeable, but Usable Gains and the Role of Hardware in Threshold Detection

Bjørn Nødland Fuglestad

Applied Computer Science
Submission date: June 2018
Supervisor: Simon McCallum, IDI

Norwegian University of Science and Technology
Department of Computer Science

Preface

This is a master thesis in Applied computer science at NTNU Gjøvik. The master thesis was performed during the spring semester of 2018. The project came to be as is after several discussions with the supervisor. Redirected walking was first brought up during a discussion regarding if hardware such as omni-directional treadmills would become the norm in VR locomotion. From then the author did some research and found that looking into the affects of hardware on the detection thresholds in redirected walking could be a possible master thesis. Then during another meeting discussing the implementation of a trial project the idea of looking in to if motion sickness was caused the moment the redirection became noticeable was suggested by the supervisor. Which is how the topic of the master thesis came to be. Now the master thesis is written assuming the reader has some experience within programming, but not necessarily within VR.

01-06-2018

Acknowledgment

The author would like to thank all the volunteers for having the time to participate in the experiment, and for enduring the simulator sickness that happened when testing rotation gain. I would also like to extend my gratitude to my supervisor Simon McCallum for the support on the quality this master thesis, and for his support over the last 5 years. I wish him luck in his future endeavours in New Zealand. I would also like to thank my fellow master students for being around to discuss our common problems and predicaments.

B.F.

Abstract

Redirected Walking uses rotation, translation and curvature gains to manipulate users of VR environments. This paper presents a new detection threshold for rotation, translation and curvature gain for the HTC Vive. The calculated rotation gain was 13% increased and 21% decreased rotation. For translation gain, 19% increased and 5% decreased movement. The detection threshold for curvature gain was a circle with a radius of less than 22m. When comparing these results to previous studies, which used hardware with lower resolution, frame rate and FOV, the results indicate that it is easier to detect two type of gain, increased rotation and decreased translation gain in the HTC Vive. Further I investigate if there is a range of gains that is noticeable, but is still comfortable for the user. The results indicate that for most, there are detectable gains that are still comfortable when noticeable. Though how great the difference in gain is between noticeable and uncomfortable is extremely dependant on the person's sensitivity to the given gain.

Contents

List of Figures

List of Tables

1 Introduction

With virtual reality headsets supporting room scale tracking becoming more and more available to the general public through systems like the Oculus Rift and HTC Vive. Research into locomotion methods for those systems is becoming more important for the future development of applications such as games and VR experiences. With several locomotion devices such as omnidirectional treadmills being developed for this purpose. In addition, with room-scale tracking, a large variety of locomotion interfaces become possible such as teleporting, natural waking, Seven league boots [1], and flying just to mention a few. As many locomotion methods have been developed, research has been done to compare the benefits of the different locomotion methods. Based on that research, locomotion methods that rely on a natural walking locomotion interface performed the better in regards to presence [2], accuracy [3], and navigation [4, 5] than several other locomotion methods. However, the downside of using natural walking locomotion interface is that it does not allow for the exploration of a virtual space larger than the tracked space. A locomotion interface that solves this is redirected walking [6] as it allows for the exploration of an infinite virtual world while keeping the same features of natural walking. In Redirected walking gains are used to create a difference between the physical transform and the virtual transform. The most common gains being rotation, translation and curvature gain. An algorithm such as steer to center or steer to orbit are then used to calculate how much gain to apply within the detection thresholds [7]. However, the major weakness of redirected walking is the large area that is required for it to function without having to apply redirection methods such as 2:1 reset [8]. As such a large focus in redirected walking research is to find ways of reducing the needed space. One of the approaches taken is to improve the detection thresholds so that more redirection can be applied. As such multiple studies into conditions to affect the detection thresholds have been performed, as seen in Tables 1, 2 and 3. However, none of the studies looked into the effect of changing hardware. Even though some studies mentioned it's possible effects on detection threshold measurements [9, 10].

With this in mind, I propose the following null hypothesis:

- H1: Improving the specifications of HMD hardware will have a no effect on the detection thresholds of rotation, translation, and curvature gain.

Given that the specification of HMD's will continue to improve, it is important to

how this will affect detection thresholds. As knowing if the detection threshold improves would allow for further optimization and if the detection threshold decreases allow for the development of countermeasures. As a second null hypothesis I propose the following:

- H2: Upon the gain becoming noticeable the experience will be uncomfortable and will remain uncomfortable for any further gain increase.

As using the detection threshold as the limit for how much gain to apply is only considered best practice to avoid simulation sickness, being able to use any additional gain would be to the benefit of reducing the space requite for redirected walking.

To test the first hypothesis this thesis will recalculate the detection thresholds of rotation, translation and curvature gain and compare the results with those of earlier studies with lower head-mounted display (HMD) specification and evaluate any changes caused by the hardware. To test the second hypothesis an experiment where the gain would increase until participants found it noticeable and then continued until it became uncomfortable was implemented.

1.1 Terminology

- VR - Virtual reality
  - Virtual world experienced through a head-mounted display.
- HMD - Head-mounted display
  - Is a device equipped on the head or as part of a helmet, that has a small display in front of each eye.
- FOV - Field of view
  - Amount of vision either horizontal or vertical commonly given in degrees
- SSQ - Simulator sickness questionnaire
  - A specialized questionnaire with the aim to quantify simulator sickness.
- 2AFC - 2 alternative forced choice.
  - A method of measuring an individual's experience. Works by presenting the individual with two options. With one of the options containing the target stimuli. The individual then has to chose the correct option.

2 Background

Locomotion in VR has been in development for several decades. With several different locomotion devices and interfaces being developed. This includes devices such as omnidirectional thread mills, pedaling devices, and foot platforms [11]. As for locomotion interfaces, a larger verity exists such as flying [2], natural walking, redirected walking [6], walking-in-place, gestures, teleportation [12], Seven league boots [1], World-in-Miniature (WIM) [13, 14], keyboard and joystick just to mention a few. In the current environment, locomotion interfaces are more common than locomotion devices as locomotion interfaces can be used by default with head mounted display's (HMD) such as HTC Vive, Oculus Rift, and PlayStation VR. The reason that locomotion devices are not that common is that they are too expensive for the average consumer as most are currently in a prototype stage and not targeted for the general audience yet.

With many locomotion devices and interfaces developed a reasonable amount of research has gone into comparing the locomotion interfaces to find what benefits each provides over the other. Such is the case with Ruddle et al.[4] which look into navigation accuracy of real walking locomotion with HMD compared to HMD with keyboard and computer screen with a keyboard. In their study, the participants would find targets hidden inside boxes in a room-sized virtual space. A real-world test with the same condition had also been done. When comparing the accuracy of the locomotion interfaces to those done in the real world the results was that with while walking with an HMD a 90% accuracy was achieved but less than 50% when using an HMD and keyboard and computer screen and keyboard. Similarly, Peck et al. [5] look into the effect of the locomotion interface on users' cognitive performance on navigation and wayfinding. Though in their study the locomotion interfaces used were redirected free exploration with distractors (RFED), walking-in-place and joystick interface. Based on their finding they concluded what RFED was superior to both walking-in-place and the joystick locomotion that they tested for the conditions. Slightly different analysis of locomotion interfaces and locomotion devices have also been done on speed and accuracy. In their study Nabiyouni et al. [3] compared a Virtusphere device with real walking and game controller locomotion. The results they found indicated that the Virtusphere was significantly slower and less accurate than both of the other locomotion interfaces, while natural walking had the best performance. A study by Usoh et al. [2] evaluates different locomotion interfaces in terms of how they affect presence in the virtual environ-

ment. In their study, the locomotion interfaces evaluated were natural walking, walking-in-place, and flying. Their results found that real walking is better than walking-in-place and flying locomotion in terms of presence.

As these studies show locomotion interfaces based on natural walking have several advantages over other locomotion interfaces and devices. As such it is likely to be the most preferred method of locomotion when possible. However, the major limitation of natural walking is that the user cannot move beyond the area of the tracked space in the virtual world. This is the main reason that other locomotion interfaces are often used to move the tracked space in the virtual world such as teleporting and flying. One of the possible solutions to this is redirected walking which allows for the movement of the virtual space using natural walking. As such it is possible that redirected walking will become a major locomotion method for those applications that require exploring a large virtual space.

The inception of redirected walking started with Sharif Razzaque [6]. With the theory of redirected walking being based on the fact that our vision often dominates our other senses. As such, when using an HMD are aware of their current movement but have difficulty visualizing the path traveled [15]. Because of this, inconsistencies can be applied to the user while moving, which makes it possible to move them down a path in the real world that is different from the virtual world. These inconsistencies are what are referred to as gains. The most common being rotation, translation, and curvature gain, as seen in Figure 1,with bending gains being more recently introduced [16].

Rotation Gain

Rotation gain is used to change the amount of rotation that is applied in the virtual world compared to the real world. Rotation gain (gR) is calculated as virtual rotation ($R_{virtual}$) divided by real rotation (R_{real}) i.e. $gR = \frac{R_{virtual}}{R_{real}}$. To apply rotation gain when a real rotation (r) happens the rotation is multiplied by the rotation gain and the new rotation is applied to the virtual camera instead. As such in the case that the rotation gain is 1 no difference in the rotation will happen. In the case that rotation gain is 0.5 the user would have to rotate 90° in the physical world to rotate 45° in the virtual world. Opposite with a rotation gain of 2 a 90° real rotation would cause a 180° virtual rotation.

Translation Gina

Translation gain is used to change the amount of movement that is applied in the virtual world compared to the virtual world. Translation gain (gT) is calculated as the change in virtual position ($T_{virtual}$), where change in virtual position is the vector (T) gotten from taking current position (P_{cur})

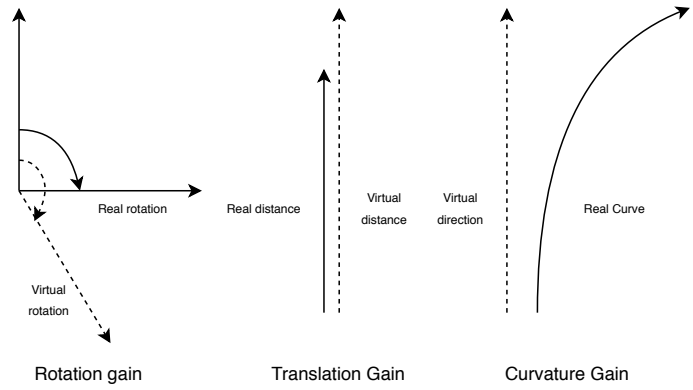

Figure 1: Illustration of the 3 main gains rotation, translation and curvature gain

minus last position (P_{pre}), divided by the change in real position (T_{real}) e.i. $gT = \frac{\delta T_{virtual}}{\delta T_{real}}$. To apply translation gain when a position change happens in the real world the vector describing the change is multiplied by the translation gain. This results in a new vector that is then applied to the virtual camera. As such when a translation gain value of 1 is applied no difference is movement is applied. If the translation gain is 0.5 waling 5m in the real world the equivalent of walking 2.5m in the virtual world. With a translation gain of 2 walking 5m in the physical world would be equivalent to walking 10m in the virtual world.

Curvarure Gain

Curvature gain is used to change the direction the participants move in the reals world while walking in a straight line in the virtual world. This possible by applying curving to the virtual camera as the user moves. To maintain a straight line in the virtual world the user is required to curve in the opposite direction. As such when they walk the path becomes curved. The curve that the user's walk in is a circular arc with radius r. From this curvature gain (gC) is defined as $gC = \frac{1}{r}$. When no curvature gain is applied ($gC = 0$) the user walks in a normal straight line. In this case, the radius would be infinite causing the curvature gain to be 0. In the case that the curvature gain is $\frac{\pi}{30}$ the user would rotate by 30° after 5m.

With gains as the foundation for redirected walking, steering algorithms are then used to apply the amount of gain within detection threshold limits. The detection threshold is the gain intensity at which any higher gain becomes noticeable.

The algorithms used can be divided into two categories *reactive* and *predictive* [17]. Reactive algorithms use a particular heuristic to make the optimal choice of gain based on the current user state. In Razzaque thesis [6] the first reactive redirection algorithms are introduced which feature steer to center, steer to orbit and steer to target. Predictive algorithms on the other hand work by predicting the user's path and optimizing the redirection based on the calculated path. Examples of predictive algorithms include FORCE [18] and MPCRed [19]. Predictive algorithms have better performance than their reactive counterparts in reducing the amount resets [17] (i.e. the number of times the user hit the bounds of the track space) that occur during play. The downside of predictive algorithms, however, is that they are significantly more complex to implement [17] than reactive. When a reset occurs during play a reorientation technique is used. Reorientation techniques where first introduced by Williams et al [8] in 2007. Williams et al. [8] proposed three different reorientation technique's: Freeze-backup, Freeze-turn and 2:1-turn. With the 2:1-turn being the most common, it uses rotation gain to double the rotation of the subject during the reset. This means that when they perform the reset they turn 180 degrees in the physical world while turning 360 degrees in the virtual world.

With these core principles in place, most research in redirected waking focuses on how to make redirected walking more usable. This includes finding what factors influence the detection threshold such as speed [20], gender [21], texture and lighting [22], sound [23], and more. These factors can then be used to improve how much gain can be used. Such as with movement speed affecting how noticeable curvature gain is. It is then possible to adjust the amount of curvature gain based on the user's movement speed. Which allows for more curvature gain to be applied than normal at slower speeds. Using illusion such as change blindness [24], four-stroke motion and motion-without-movement have also been shown to be efficient ways of improving redirected walking [25, 26]. In a similar vain eye tracking is a feature that is showing promise in improving redirected walking. One way eye tracking is used in redirected walking is in predictive algorithms as gaze can be used to predict the user's locomotion target [27]. Another usage for eye tracking in redirected waking is to make it hidden from the user by applying the gains during saccadic suppression [28]. Other research focusing on how to make redirected walking usable in a multi-user environment. Which involves developing new redirected walking to algorithms [29] to prevent the users in the same space from colliding.

2.1 Related Work

In regards to related work concerning the second hypothesis of gains above the detection threshold, there has been little research within that area. As most have

focused on the detection threshold because sub-threshold gains do not lead to any increases in simulator sickness [30]. While that is the case gains beyond the detection threshold has been used in redirected walking in relation to 2:1 resets [8]. However in the Study by Williams et al. [8] simulator sickness is not mentioned. As such one can only speculate if any discomfort was felt during those experiments. However given that the 2:1 reset was their preferred reset method the effects could not have been too extreme. In addition, the concepts are mentioned as future work in a recent study [31] on the field. As such, it suggests that the idea has not been investigated.

In regards to the first hypothesis, finding papers on the detection threshold was not an issue. As gains are a fundamental part of redirected walking there have been several studies on the detection thresholds for the three main gains looking into what factors affect it. As such 3 tables have been compiled containing the detection thresholds found in the studies and the hardware use. Similar tables have also been created by Langbehn et al. [31] containing an overview of detection thresholds including bending gains. The detection threshold measured in the studies in Table 1 were all for rotation gain along the horizontal axis (yaw). The reason for only applying rotation gain along yaw is because applying rotation gain to pitch and roll can't be used to reduce turn circles when performing redirected walking. In regards to the translation gain, the detection threshold measured in the studies in Table 2 are all done with walking in mind. As such the detection thresholds calculated do not apply to any sideways movement (strafing), even though it is possible to apply translation gain to sideways movement. It is also possible to apply translation gain to vertical movement (jumping) however this is not useful from a redirected walking perspective as it cannot be used to influence the position in the tracked space. With curvature gain, the detection thresholds in Table 3 are also calculated with walking in mind, just like translation gain.

For the majority of these experiments, the method used to measure the detection thresholds was by using two alternatives forced choice (2AFC), though it would be more accurate to call it a pseudo-2AFC task according to Grechkin et al. [10]. As a true 2AFC task in psychometric literature would give the participant two different stimuli before having the participants chose one of the two alternatives. The modified version that is commonly used in redirected walking on the other hand only provides one stimulus at a time to the participant and then having the participant chose one of the two alternatives. As such it would not be true to call it a 2AFC, but rather a variation of a yes/no task. The reason that it is not just a yes/no task is that the two alternatives are not presented as a yes/no question to avoid bias. Because of this, it will be referred to as pseudo-2AFC from here on as suggested by Grechkin et al. [10]. The reason for why a pseudo-2AFC is used over a

true 2AFC task is mainly because of the nature of the stimuli. As testing gains take considerable time given the physical exertion involved. In the case of Steinicke et al. [7] completing their experiment took on average 3 hours per participant. As such having two stimuli instead of one could potentially double the time and effort needed to complete the experiment, making testing increasingly difficult. Because of this, using the pseudo-2AFC has been the most common approach.

The method for which stimuli is tested is also a important component of the pseudo-2AFC. The most common approach used when calculating detection thresholds in redirected walking is to use the method of constant stimuli [10]. With the method of constant stimuli a range of gains are used as the stimuli, with each stimuli being tested a given amount of iterations in random order. For each gain the probability of a "correct" response can then be calculated. A psychometric function (sigmoid function) is then fitted to that data. The function then describes the probability of detecting any given gain in it's range. The gain were the probability is 50% is the point of subjective equality (PSE). The PSE is the point were the two stimuli are experienced as equal. It can also be used to evaluate response bias by comparing the value to the point of objective equality (POE). The gain values at 25% and 75% are then the lower and upper detection thresholds. In the case that a the task was a regular yes/no task the detection threshold would then be the gain at 50% probability. The disadvantages of using the method of constant stimuli as pointed out by Grechkin et al. [10] is that it requires a large number to iterations for each gain to get sufficient accuracy. Such as, in Steinicke et al. [7] when testing rotation gains a range of 10 gains were tested with 10 iterations meaning a 100 tests were performed per participant. Which is large reason why calculating detection thresholds like this take a large amount of time. As an alternative Grechkin et al. [10] suggest adaptive methods as a possible alliterative to method of constant stimuli. As adaptive methods uses the previous responses to calculate the next stimuli. This method reduces the number of test needed to calculate the detection threshold. Though this it is more susceptible to bias. In addition, Grechkin et al. [10] results show that the detection thresholds are affected by the change to estimation methods when comparing method of constant stimuli and adaptive methods. As can be seen in the difference between the measured detection thresholds in their first and second experiment 3.

Changes to the calculated detection threshold can also be seen in Steinicke et al. [32] where the task to perform was changed from experiment 1 to 2 3. This goes to show that the task performed during the pseudo-2AFC will also have an effect on the measured detection threshold. Another effect on the detection threshold at least in rotation gain is visual density. Paludan et al. [33] in their study found that at least 4 objects are required to get an accurate measure of detection threshold for rotation

gain. In the case of 0 objects in the virtual environment, participants were not able to distinguish if rotation gain was applied or not. It would be reasonable to assume that visual density would have a similar effect on curvature and translation gain. As without any points of reference participants would be unable to notice movement or acceleration. Another variable that effects detection thresholds in curvature gain is movement speed [20, 34]. With slower movement speeds allowing for more curvature gain to be applied. The same could be true for translation and rotation gain alas it has not been measured. Other factors that have been shown to effect detection thresholds are participant gender [21] and audio [23].

While each of the different studies provides information regarding factors that affect the detection threshold no study measures what effects changes in hardware cause. However, it has been noted as a possible variable when measuring the detection threshold in previous studies.

"Furthermore, we believe that view angle of the HMD is an important factor in the amount of immersion that can be achieved, correlating with the users' trust in the virtual scene"[9]. "In our experiment, the field-of-view was significantly larger. Because peripheral vision plays an important role in motion detection, these technology differences could have affected the estimates."[10].

When looking to compare previous studies on detection threshold for the effect of hardware changes the most relevant studies that use low specification hardware would be Steinicke et al. [32, 7] studies from 2008 and 2010. As they covers the 3 main gains and with Steinicke et al. [7] study from 2010 being the most cited paper regarding detection thresholds in redirected walking. Both of the studies uses pseudo-2AFC with method of constant stimuli as estimation method. With 10 iterations for rotation gain and 8 iterations for translation and curvature gain making them some of the studies with the highest measurement accuracy. Bruder et al. [21] study from 2009 is also a relevant low spec study as it covers the same areas a the two before motioned studies. However, the detection thresholds are not as reliable because the detection thresholds are split between male and female with few participants for each. The other studies that also cover all gain Bruder et al. [35] study from 2012 and another Steinicke et al. [36] from 2008 have other issues that make them unusable when evaluating hardware changes. Such as the measurements in Steinicke et al. [36] were done with two different HMD. In the case of Bruder et al. [35] the amount of iterations done for each gain were only 4. As such the accuracy of the calculated thresholds is questionable. The other studies for rotation gain in Table1 all have issues that make them bad for evaluation hardware changes. In the case of Engel et al.[9] the HMD used is not specified. Chen at al. [37] and Paludan et al.[33] have the same issue as Bruder et al. [35] study from 2012 where each gain was only tested 4 times. As for for Meyer et al.[23] their study has audio

redirection in addition to visual redirection as such cannot be compared to other detection thresholds. The detection thresholds found by Grechkin et al. [10] for curvature gain with no translation gain applied could possibly be used, however the task performed for their first experiment is not the same as those by Steinicke et al. [32, 7] fist mentioned studies and Bruder et al. [21] study from 2009. Meaning that the detection threshold is likely affected. In the case of their second experiment an adaptive method was used instead of a method of constant stimuli. Given that they found that the method used changes the detection threshold using these results to compare hardware changes with those of Steniche et al [32, 7] and Bruder et al. [21] study from 2009 would not be possible either. In the case of Niera et al. [34] the results were from testing with a wheelchair and with Neth et al. [20] The effect of walking speed was tested. In the case of bending gains, as it was only recently created no studies other than the original by Langbehn et al. [16] have calculated any detection thresholds for bending gains. As such, because of the variables that influence the detection thresholds comparing the results between the studies with old equipment and new equipment to find the effects of hardware changes is not possible. However, the studies by Steinicke et al. [32, 7] would be the most accurate to compare any new results with as these studies with had the most amount of iterations among the studies with low specification for their HMD.

Paper	HMD			Rotation gain			
	Res	Frame rate	FOV	Inc (+%)	Dec (-%)	PSE	Participants
Steinicke et al, 2010[7]	800 x 600	60Hz	40	49	20	0.95	13
Steinicke et al, 2008[32]	800 x 600	60Hz	40	68	10	0.8403	12
Engel et al, 2008[9]	?	?	?	35	15	~1.12	10
Chen et al, 2014[37]	1280 x 800	90Hz	110	15.9	9.7	?	10
Bruder et al, 2009[21] (Male)	800 x 600	60Hz	40	44	16	0.9447	7
Bruder et al, 2009[21] (Female)	800 x 600	60Hz	40	51	21	0.9642	6
Meyer et al, 2016[23]	1280 x 800	90Hz	110	36	32	?	20
Paludan et al, 2016[33]	1280 x 1024	60Hz	60	19	19	1	17
Bruder et al, 2012[35] (Wheelchair)	1280 x 1024	60Hz	60	26.2	22.81	1.0111	12
Bruder et al, 2012[35] (Walking)	1280 x 1024	60Hz	60	25.94	31.9	0.9544	12
Steinicke et al, 2008[36]	1240 x 1024 800 x 600	60Hz 60Hz	80 40-45	30	30	-	8

Table 1: Overview of study reporting detection thresholds for rotation gain and the hardware used.

Paper	HMD			Translation gain			
	Res	Frame rate	FOV	Inc (+%)	Dec (-%)	PSE	Participants
Steinicke et al, 2010[7]	800 x 600	60Hz	40	13	26	1.07	12
Steinicke et al, 2008[32]	800 x 600	60Hz	40	19	22	0.9972	14
Bruder et al, 2009[21] (Male)	800 x 600	60Hz	40	11.2	20.6	1.0776	7
Bruder et al, 2009[21] (Female)	800 x 600	60Hz	40	16.2	19.4	1.0535	6
Bruder et al, 2012[35] (Wheelchair)	1280 x 1024	60Hz	60	36.07	6.22	1.1508	12
Bruder et al, 2012[35] (Walking)	1280 x 1024	60Hz	60	28.96	12.76	1.0824	12
Steinicke et al, 2008[36]	1240 x 1024 800 x 600	60Hz 60Hz	80 40-45	45	15	-	8

Table 2: Overview of study reporting detection thresholds for translation gain and the hardware used.

Paper	HMD			Curvature gain		
	Res	Frame rate	FOV	Rad (m)	PSE	Participants
Steinicke et al, 2010[7]	800 x 600	60Hz	40	22	0.002	12
Steinicke et al, 2008[32] E1	800 x 600	60Hz	40	16	-1.74	10
Steinicke et al, 2008[32] E2	800 x 600	60Hz	40	24	-1.37	10
Bruder et al, 2009[21] (Male)	800 x 600	60Hz	40	17.4	~0	7
Bruder et al, 2009[21] (Female)	800 x 600	60Hz	40	24.9	~0	6
Meyer et al, 2016[23]	1280 x 800	90Hz	110	6	?	20
Neira et al, 2012[34] (Wheelchair)	1280 x 1024	60Hz	60	5.76 (0.33 m/s) 16.52 (0.54 m/s)	?	10
Neth et al, 2012[20]	800 x 600	60Hz	40-45	10.57 (0.75m/s) 23.75 (1.00m/s) 26.99 (1.25m/s)	~0	12
Bruder et al, 2012[35] (Wheelchair)	1280 x 1024	60Hz	60	8.97	~0	12
Bruder et al, 2012[35] (Walking)	1280 x 1024	60Hz	60	14.95	~0	12
Steinicke et al, 2008[36]	1240 x 1024 800 x 600	60Hz 60Hz	80 40-45	3.3	-	8
Grechkin et al, 2016[10] E1	960 x 1080	60Hz	110	8.54 (0.75 gT) 11.61 (1.00 gT) 15.63 (1.40 gT)	-	12
Grechkin et al, 2016[10] E2	960 x 1080	60Hz	110	6.37 (0.75 gT) 6.41 (1.00 gT) 6.85 (1.40 gT)	-	17

Table 3: Overview of study reporting detection thresholds for curvature gain and the hardware used.

3 Methods

In this chapter how experiment is performed is described. Along with information regarding the experiment conditions and setup.

3.1 Experiment Setup

The HMD used during the experiments was the first commercially available HTC Vive. The HTC Vive features 1080 x 1200 resolution for each eye, 110 degrees field of view, 90Hz refresh rate and a 5m cable. The chaperone bounds created when running the HMD with steam were removed as much as possible. As participants in the experiment could use them to observe the gain being applied. Not all of the chaperone bounds could be completely removed. It was only possible to turn off the vertical bounds for the 4 x 4 maximum play space. The floor bounds marking the edges of the tracked space could not be turned off, as seen in Figure 3. Instead, the opacity was set to as low as possible and the color of the bounds was set to white. The physical room used for the experiments was about 7m by 7m in dimensions and can be seen in Figure 2. Windows in the room where covered and the lights were turned off during experiments. Participants also wore Bose on-ear Soundlink headsets during the experiments. The computer used to run the experiment on had the following specifications. The CPU was an Inter core i7 with 2.8 GHz, it had 12 GB of ram, a Nvidia GeForce GTX 980 GPU and ran Windows 10 Pro 64-bit as its operating system. When running the experiment application had a stable 90 frame per second.

3.2 Questionnaire

Before starting the VR experiment a demographics questionnaire would first be performed. Followed by a simulation sickness questionnaire [38] (SSQ) before and after the VR experiment, the questionnaire can be found in the appendix C. The demographics questions were intended to get similar information from the user as previous studies [7, 32].

3.3 Experiments

Before starting the main experiments the participants entered a tutorial room to familiarize themselves with the controls and VR, as seen in Figure 4. After the tutorial instructions, the participants required to do a small test that measured the participant's gait and eye height. Afterward, the main experiments bellow would

Figure 2: Physical Experiment Room

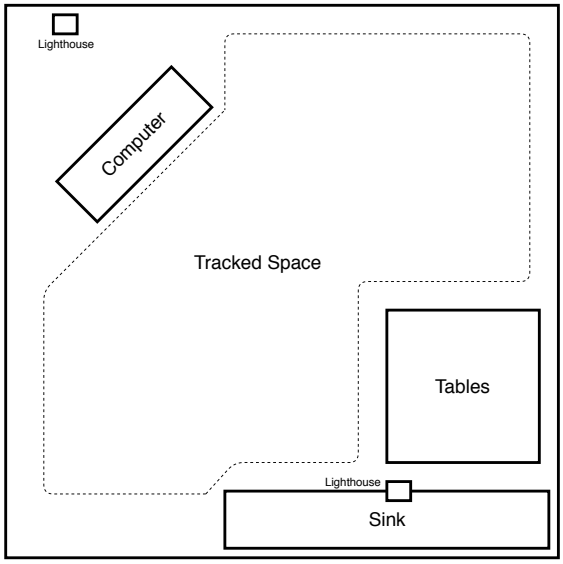

Figure 3: Illustration of tracked space in physical room

start. The instructions on how to carry out the experiment were given primarily in VR through both text and audio, the instructions can be read in the appendix B. In the case that instructions were unclear, the Author would provide additional help. Ambient sound was played to prevent participants from using any potential outside audio cues to evaluate their position. When the experiments were finished participants were offered sweets, but no other reward was given or mentioned until after completion. A participant was allowed to participate in the experiment multiple times as long as different gains were being tested. In the case of simulation sickness, the patients could take a break at any time or decide to quit the experiment.

3.3.1 E1: Incrementing Gain Experiment

The first experiment aimed at testing the second null hypothesis by testing if the VR experience becomes uncomfortable upon participants noticing the gain being applied. As such in this experiment, the gain is increased over time until it first becomes noticeable and then uncomfortable. When performing the experiment the participants would indicate using the up button on the trackpad of the HTC Vive controller when they experienced that the gain became noticeable. The next time the participants pressed the button would be when they reached a gain that they could not tolerate. The gain would increase at random between every 10 to 15 seconds. A non constant gain increase was chosen so that the participants had enough time to evaluate each gain increase. How much the gain increased each time depended on both the gain being tested and if the gain tested was increasing in a negative or positive direction. For rotation and translation gain the experiment would then repeat with the opposite gain, so that both negative and positive gains were tested. The order in which they were tested was randomized. For curvature gain only left or right curvature was tested as previous experiments have found no difference between the two [32, 35]. With translation and rotation gain the gain would change randomly between ± 0.05 to ± 0.1 gain depending on if negative or positive translation or rotation gain was being tested. Both rotation gain and translation gain would start at a gain of 1. While with curvature gain the gain would start with a curvature gain of $\pi/180$ and then 10% gain would be subtracted with each gain increase. To prevent the experiment from running indefinitely a limit was set for each gain at which the experiment would stop. The limits set for rotation and translation gain was when the gain became less than 0.1 or more than 5.0. When testing positive rotation and translation gain the gain increase was multiplied by a factor of two to reduce the time used in the experiment because of the high limit. With curvature gain, the limit was when the gain became less than $\pm\pi/8$. To encourage rotation when rotation gain was tested the participants were given a gun that could be used to shoot at disappearing targets, as can be seen

in Figure 5. With translation and curvature gain two cannons would shoot slow moving balls that the participants could catch to score points. Because of the cable length and the room size it was necessary for the examiner to follow behind the participants and gently pull the cable to prevent them from hitting walls during this experiment.

3.3.2 E2: Detection Threshold Experiment

This experiment was intended to test the first null hypothesis. This experiment was designed to be as close a match as possible to the earlier studies by Steinicke et al. [32, 7] to allow for the comparison of the results. As such a pseudo two-alternative forced-choice (2AFC) task was used with a method of constant stimuli. All the virtual rooms made for this experiment was large enough to ensure that there were no vertical objects within 10m of the starting position. The starting positions were not randomized as the participant would always start at the center of the virtual testing room. However, the relative location and rotation carried over from the experiment causing some randomness in the start position. When translation and curvature gain was tested in this experiment the participants were allowed to listen to music of their own choice to make the experiment less repetitive and boring.

Rotation Gain

For rotation gain, the task participants were required to perform consisted of rotating 90 degrees to either side and look at a green circle while a gain was applied. When the circle turned blue they were required to indicate if the virtual rotation was greater or lesser than the physical rotation. Though a simpler instruction was also added which told the participants that they could use rotation speed to make the same judgment. This was then repeated for the remaining gains. The range of rotation gain used was from 0.5 to 1.5 with increments of 0.1 in between with each increment being repeated 10 times. The virtual room used can be seen in Figure 6.

Translation Gain

Before the translation experiment could start the virtual room and physical room needed to be aligned. This was required as the room used did not provide enough space for the participant to walk freely in any direction when performing the translation gain task. As such the virtual space and physicals space was aligned so that a 1m wide path in the virtual world would match the diagonal of the room, as seen in Figure 7. This was done by the researcher. The participants were then required to walk along the path towards a green ball while a gain was applied, as seen in Figure 11. The ball would then turn blue after the participant had walked 5m in the virtual world. The participants were then required to evaluate if the virtual distance traveled was greater or lesser than the physical distance. Though

as with rotation gain a simpler instruction was also added that specified that the users could use movement speed to make the same judgment. After answering a green circle would appear on the floor behind them to indicate the reset position. After having walked back the steps above would be repeated until there were no more gains to test. The range of translation gains used was from 0.6 to 1.6 with 0.1 increments, with each increment being repeated 10 times.

Curvature Gain

Curvature gain used the same steps as translation gain. The main differences were that no gain was applied for the first 1.5m walked. Making the virtual distance required to walk before the ball turned blue was 6.5m. The virtual walking distance was reduced from 7m in previous studies [7, 32], because if 7m was used the participants would collide with the walls of the room when the curvature radius was decreased. When testing curvature gain the following range of curvature gains were used $[\frac{\pi}{180}, \frac{2\pi}{180}, \frac{3\pi}{180}, \frac{4\pi}{180}, \frac{6\pi}{180}]$ which correspond to $[5°, 10°, 15°, 20°, 30°and0°]$ scene rotation after 5m, again 10 iterations for each. For curvature gain, only positive values for the gain were used as this made the participants curve left. In the case that the participants had curved right the HTC Vive cable could be used to determine curvature by feeling the drag on the cable, as right curvature would have the participant further from the computer. As such no negative curvature gains were tested as previous work [32, 35] has shown that there is no statistically significant difference between left and right curvature. Further, the participant was required to answer if they noticed any curvature or not. Because of this and only using positive curvature gain, the method used was no longer pseudo-2AFC as used in by Steinicke et al. [7] in their 2010 study but a binary yes/no response more similar to their 2008 study [32]. Though changing to a binary yes/no response will introduce response bias, it was still done given the conditions mentioned above.

4 Implementation

This Chapter will cover the implementation of the experiments and the challenges that appeared during implementation. Because of the scale of the project, the parts of the implementation that not of great importance but are needed for replication have been moved to the appendix A.

4.1 Development environment

The software used for developing the application that runs the experiments was made in unreal engine 4 [39] version 4.18.3. The reason for using a game engine was that it provided a development environment with HMD support, with minimal development time needed to get a simple VR application working. Further, a game engine also provided the framework for 3D rendering and experiment logic that was required. As such by using a game engine the development time of the application would be significantly reduced compared to developing from scratch. As for the reason why the unreal engine was chosen over, for example, the Unity [40] engine which featured a redirected walking toolkit [17] or any other engines was because the unreal engine was the game engine that the author had the most experience with. The author believed that more time would be spent on developing the logic surrounding the experiment rather than on implementing the gains needed for testing. As such the redirected walking toolkit did not provide enough benefits for the Unity engine to chosen over the Unreal engine.

Within the unreal engine editor, visual scripting using blueprints was the primary method for implementing the code needed to create the software for the experiment. C++ was also used in some areas however blueprints were preferred as it allowed for more rapid development and testing.

4.2 Virtual environment

The initial idea for the virtual environment was to have it be something familiar to the user such a the town environment as used by Steinicke et al [7]. However, the Author was unable to find the assets that were used by them, and creating a 3D town environment from scratch was would be too time-consuming. As such the idea of using a physical scene turned virtual as the virtual environment in the experiment was abandoned. From there on the idea of using different floating platforms was considered and partially implemented but were abandoned in favor of a skyscraper setting. From there on the floors were designed to fit the specific needs

Figure 4: Tutorial Room

of the experiments. I the case of the tutorial floor, as seen in Figure 4, it was designed to have little visual information to allow the users to focus on learning the controls. In the case of the floor used for the increment experiment, as seen in Figure 5, the floor was made to contain a variety of object that the user could interact with during the experiment. The floor used for translation and curvature thresholds, as seen in Figure 7, was designed with the description of previous studies in mind. As such the starting position in the middle of the floor was 10m from any vertical object, and a 1m line was added to the floor for curvature gain as in previous studies for curvature gain. The same floor was used for both translation and curvature gain as the conditions for the tasks the users needed to perform were very similar. A separate floor was created for rotation gain that featured more objects in all direction. Though each of the floors was decorated sufficiently to provide enough visual density [33] for redirected walking to be noticeable. In addition, the overall lighting in the environment was made darker than the default light settings as recommended for VR in the Unreal engine [41].

4.3 User Interface

The user interfaces used in the experiment consisting of a 2D menu for experiment settings and a 3D menu used for instructions and experiment progression. The 2D menu was displayed at the start of the experiment and was used to specify the participant id, which gain to test, FOV and where to save the data, as seen in Figure 8. The FOV option was included in the case that there was time to test other FOV's however as there was not these have been grayed out. The reason for using

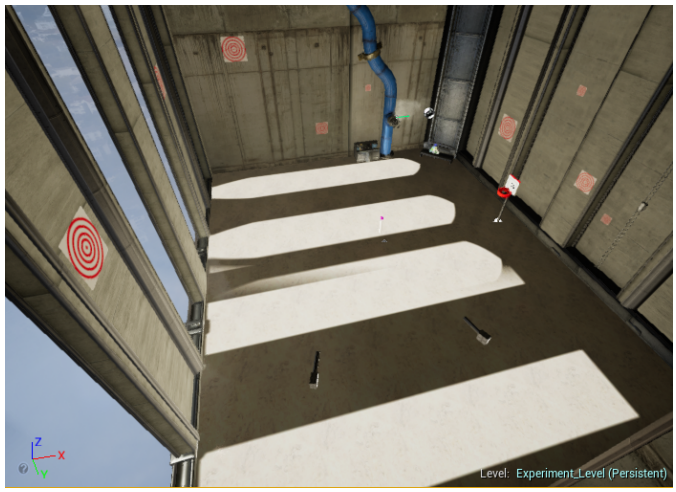

Figure 5: Incrementing gain experiment room

Figure 6: Rotation gain experiment room

Figure 7: Translation and curvature gain experiment room

a 2D menu instead of a 3D menu for this was because of functionality. As when working with 3D menus in VR the interaction is for the most part limited to just button pressing. Using the keyboard while in VR was tried but the author was not able to get it working. As for using a 3D keyboard, while it was a possible option making a 2D menu was far more time efficient.

For displaying the instructions to the participants a 3D menu was created. It then connected to the virtual model of the left controller, as seen in Figure 9. The reason for attaching the 3D menu to the hand was because it allowed the user to zoom in and out as they pleased, compared to static 3D menus that the Author had experimented with before. With static 3D menu how clear the text was would often depend on where the player was standing and there was a bigger risk of other object getting in the way. In addition to displaying the instructions to the participant the menu also allows them to mute or un-mute the voice instructions or repeat them if needed. Last, the 3D menu was also used to control the progression of the experiment. As a begin button was available at the start of the experiment. Then whenever a new instruction was given the experiment would pause until the participant pressed the continue button. Finally, when the experiment stage was complete a finished button appeared that would take the participant to the next stage of the experiment when pressed.

4.4 Game Instance

The game instance class in the Unreal engine is a class that creates an object that will remain in memory for as long as the application is running. Because the 2D

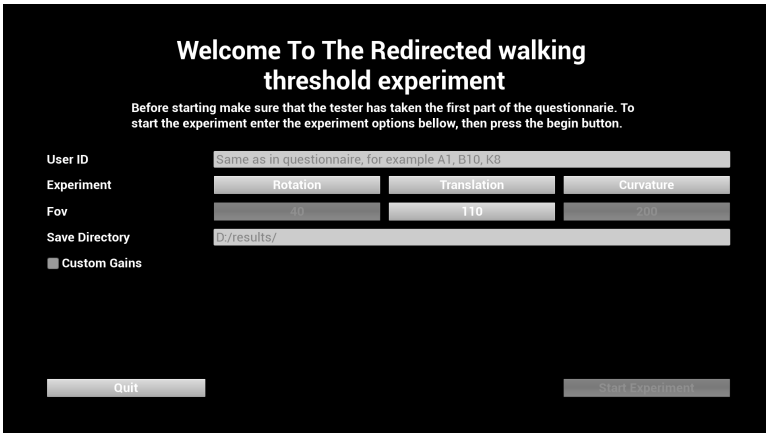

Figure 8: 2D experiment options menu

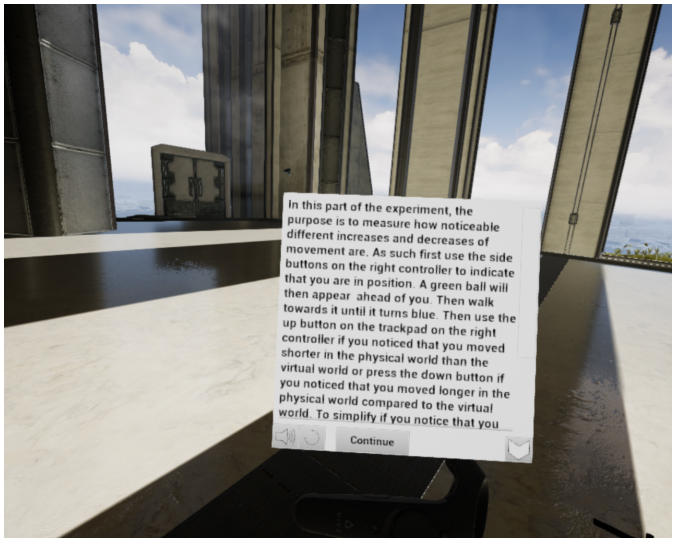

Figure 9: 3D menu for displaying instruction and experiment progress

menu used for starting the experiment was on its own level while the experiments are on another, if the data entered was not stored outside the level it would be lost when switching level. As such the game instance class was used to store the data entered in the menu when the experiment was started and all the data that was collected afterward. The data stored in the game instance object consists of:

- UserId
  - Participant id entered during start menu.
- FOV
  - Which field of view was selected from the start menu.
- Experiment type
  - The experiment type entered from the start menu. Refers to either rotation, translation or curvature gain.
- Eye Height
  - The eye height measured before the first main experiment.
- Gait
  - The gait measured before the first main experiment.
- Detection Positive
  - The gain value that the participant first detected the positive gain.
- Detection Negative
  - The gain value that the participant first detected the negative gain.
- Max Gain Positive
  - The positive gain value that the participant no longer felt comfortable with.
- Max Gain Negative
  - The negative gain value that the participant no longer felt comfortable with.
- Mapped Threshold Answers
  - The answers during the threshold experiment added to a map so to be easier to summarized. To ensure that the gain were printed in the correct order the gain key values used would be added in order when the threshold experiment started. As this was found to be a simpler work around than having to sort the keys afterwards.

- Serialized Thresholds Answers
    - The answers to the threshold experiment in the order they were given.

Most of the data was public so it was easy for the VR Pawn class, see Section4.6, to add the results to the object instance. However, the threshold answers had their own function for adding the results. This was because a Boolean was used to determine if the experiment task had been completed and that the answers were being accepted. The Boolean was also used to prevent accidental answers or multiple answers at a time.

In addition to storing the data the game instance class was also responsible for saving data. The save and load text from file functionality needed to be implemented in C++ and exposed to blueprints as those functions were not available when using blueprints. The code used to perform save and load from file were from Rama's Unreal engine Victory plugin [42]. From the data in the game instance class a total of three files were created. The first file contained all the information gathered on the participant running the current experiment. The other two files were summary files were the results from this experiment were appended to. One of these files was a summary of each participants UserId, Eye Height, Gait, Detection and Max gain values. The last was a summary of the mapped threshold data. All the data files were save in a comma separated format to be easily added to spreadsheets or easily included the thesis. To convert the stored data to the correct format functions for turning the data to a string was added. In addition, to prevent the results from overriding each other each of the files contained the field of view and experiment type in the name. For the file containing all the participant data the UserId was also added to the file name. In the case that files already contained data the data would always be appended. Further to prevent data loss the save function was called after each experiment even though it caused some data duplicates in the summarized files that would then manually removed. Last the game instance class was also used to save the tracking data collected however it did not store the tracking data as that was stored in the VR Pawn 4.6. Initially the tracking data was converted to a string by appending every piece of data to the same string. However during the initial experiments these caused the experiment to freeze when the tracking data was saved, as the append operation could take several minutes depending on the amount of data to save. As such instead of appending the data each set of tracking data was converted to string separately and stored in a new array. The engine then provided a function for merging the array of strings to a single string with a separator. When this was implemented the wait time for saving the data was no longer noticeable.

4.5 Level and Game Mode

In unreal engine levels are different 3D spaces each containing their own assets and rules. In total 2 levels were created in this project. One for containing the 2D starting menu, the other for running the experiments. The only functionality of the level containing the 2D staging menu was to display the menu and show the mouse cursor. The experiment level starts by enabling the HMD as it was not started when the application launches. In addition, it provides the different start positions of the floors to the Game Mode. In the Unreal engine, the Game Mode class is commonly used to control the rules within a level. The Game Mode used in the experiment level was used to transition between the experiments by spawning the correct subclass of VR_Pawn on the correct floor depending on the gain and experiment being tested.

4.6 Pawn

In the unreal engine, a pawn class is used to create an object that is controlled by the player which is not humanoid and does not use humanoid movement. As the HMD only provides references to hands and head this was used as the base class for the user-controlled object. The implementation of the VR pawn was distributed over several subclasses to separate functionality. A class overview can be seen in the Figure 10.

VR Pawn

Starting with the base class the VR_Pawn, this class was designed as a standard VR pawn without any redirected walking or experiment related code. To starts with the class uses a camera component for head tracking. The camera component was attached to the root scene component and offset by -100 in the z-axis. The offset was required to make it so that the height from the virtual floor to the camera component feels the same as it is from the real floor to the users head. The lock to HMD option was also set on the camera component, though in theory it was not required as the camera would follow the HMD even if this option was turned off. Next the pawn has two motion controller components, one for each hand. These provide the tracking needed for hand movement, however do not need to be offset like the camera component. Each of the motion controller components has several subcomponents. This includes a skeletal mesh, several text render components and a sphere collision. The skeletal mesh mentioned was for displaying a 3d model of the HTC Vive controller. The text render components are used for displaying the what functionality the buttons on the controllers have. The sphere collision components are used for grabbing overlapping object with the hands by attaching them to the actor when the HTC Vive controller's trigger was pressed and dropping

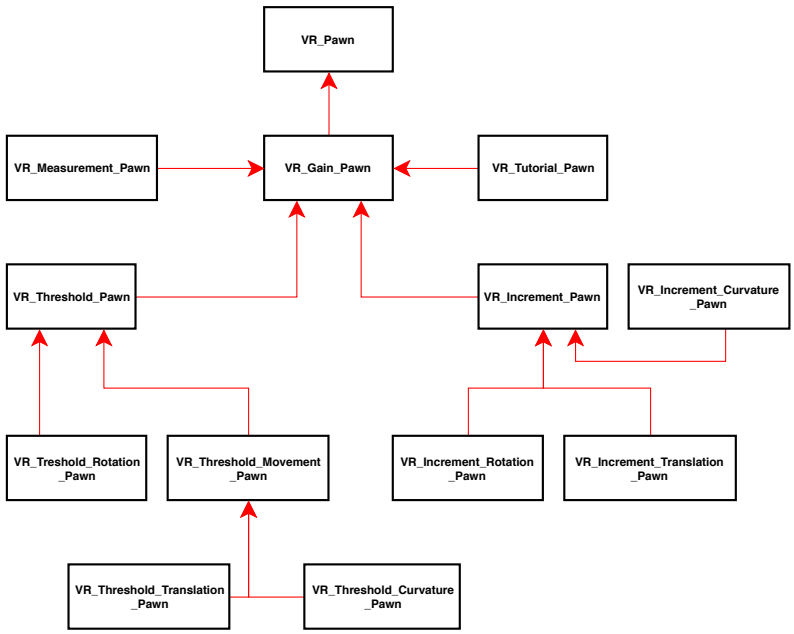

Figure 10: Pawn class inheritance diagram

them when it was released. In addition the right controller contains an additional static cylinder mesh and a widget interaction component. The widget interaction component was used for interacting with 3D menus as it functions as a mouse cursor. Though while the widget interaction component has its own debug line for showing where it points, this line was found to big with the 3D menu used. Because of this the cylinder mesh was used to show where the widget interaction component pointed instead. The pawn also handles some input such as grabbing and dropping as mentioned before. Other inputs included toggling the visibility of the text render components and cylinder mesh through the side button and the menu button on the left controller respectively. It also handled two keyboard inputs that the Author could use during the experiments. This includes the escape key used to exit the application and the S key which was used to force save data. Last when it was no longer controlled by the player it automatically removed itself.

VR Gain Pawn

This sub-class of the VR_Pawn handles 3 things. First, it gathered movement data and calls the functions used to apply the gains, however it did not apply any of the gains itself. The data gathered for gains to be applied was the changes in relative position and rotation of the camera component. As the camera component was controlled through the HMD it's relative position and rotation matches that of the physical movement of the user. The changes in position and rotation are calculated at each tick. The change in the rotation was calculated as the signed angle between the current forward vector and the previous. The z-axis in the forward vectors was set to zero to make sure the singed angle only reflects the rotation along the z-axis (yaw). The delta movement was calculated as the difference between current and previous position, but again only along the x and y-axis. Having calculated the delta rotation and movement an "applying gain" function was called if the "should apply gain" function was true. The "apply gain" function was not implemented in this pawn and but was overridden by sub-classes. How this function was implemented in the sub-classes for the different gains is described in section 4.6.1. The "should apply gain" function was overridden by sub-classes as well as each has different needs for when gain should be applied.

Second, the VR gain pawn handled collecting the tracking data from the camera component. The data collection was done after the gain was calculated in the tick function. The tracking data would be gathered as long as the experiment was not paused and as long as a certain amount of time had passed. This was because a sample rate had been added as it was not necessary to sample every frame. A high sample rate of 40 samples per second was normally used however, it could be changed depending on the circumstance. Such as when the application was paused or the experiment had not started yet. If the collection was done the tracking data

containing the user's current camera position and rotation would be added to a struct along with a string to specify the tracked data type. The struct would then be added to an array. Other than position and rotation data, events such as gain change and results from experiments were also added to the tracking data. In the case that the array exceeded 150000 entries it would be automatically saved to file. This was initially added to prevent too much memory from being used but was actually never needed.

Third, the VR gain pawn handles changing the instructions, playing audio instructions, and stopping them. The 3D menu used to give written instructions was added to the pawn in this class. The widget (unreal name for GUI elements) component was added as sub-component of the left-hand motion controller. In addition to the widget component and audio component was added as well to the left hand to play the audio instructions. The instructions themselves were stored in an array of structs, with each struct containing the written instructions and it audio pair. This array was filled during the construction of the other sub-classes allowing each of them to customize their own instructions. To move through the instructions a "next instruction" function was used. When this function was executed was decided by the sub-classes. What the function does was change the text currently being displayed in the 3D menu as well as the buttons if needed. It would also play the audio instructions that were paired with the text instructions. In the case that it was the last instruction, the finished button would be displayed in the 3D menu.

An interesting observation made when performing the experiments was that throwing objects that had been picked up was no longer possible when the gain was applied. In the case that an object is thrown while the gain is applied the object will fall straight down to the floor. A speculation as to why this is the case is that when the gain offset is added to the actor the momentum of the grabbed object is reset. As this did not create any problems for running the experiment it was not attempted to fix. However, it does strike some questions regarding how redirected walking should be implemented in the unreal engine or if how objects are picked up in VR needs to change.

VR Measurement Pawn

This sub-class was used to measure the gait and height of the participants. When the participant pressed the top button on the trackpad of the right controller the position current camera position would be stored. When they released the button after presumably having taken two steps the distance between the starting position and the current position was calculated and divided by two to get the gait. To record the participant's eye height the Z value of the location of the camera component was recorded when the down button on the trackpad of the right controller was pressed

VR Increment Pawn

The increment pawn was the sub-pawn responsible for handling the logic needed for the experiment with increasing gain. For the increasing gain experiment to work this pawn handles increasing the gain at random intervals, getting the answers and progressing the experiment. Variables such as how much the gain was going to increase, the max and minimum gain values, time intervals minimum and maximum range and positive and negative gain multiplier are variables in this class that are set by the gain sub-classes to fit their requirements. To increase the gain the pawn when enabled by starting the experiment. The delta time was then aggregated in the tick function. When the aggregated time was more than the time interval a new gain was gotten from a "gain increase" function, the new gain was recorded in the tracking data, a new time interval was set, and the time aggregated was set to zero. To ascertain if the new gain was within valid bounds a Boolean was gotten from a "get within bounds" function. In the case that new gain was outside the bounds of the gain tested the experiment would then auto progresses. When auto progressing same happens as if the participant found a uncomfortable gain. To signal that the gain becomes noticeable or uncomfortable the up button on the right trackpad was used in this class, as mentioned in the Method Chapter 3. If the pawn was enabled when pressed the function triggered would check if this was the first time the button was pressed. In the case that it was the first time the current gain was recorded as the detected gain, if not the current gain was recorded as the uncomfortable gain. Afterwards the "next instruction" function was called. The first time a uncomfortable gain was reached the direction of the gain being tested was then reversed, and the current gain reset back to the starting value. Boolean's are also adjusted so that the next time the button was pressed the gain applied was recorded as a detected gain again. In the case that both positive and negative gains have been tested the the last instruction would be displayed, along with the finish button for ending the experiment. Though in the case of the curvature gain subclass the Booleans were set so that the experiment was half completed, such as to only do one measurement.

For the gain sub-classes of the increment pawn, each adjusts the variables, override the "increase gain", "bound function" and "apply gain" functions to meet their needs. The values for the variables that each gain use are described in the Method Chapter 3.3.1. The only big difference between these three gain pawn classes was that the rotation pawn features a gun for shooting. The gun itself was a skeletal mesh that was attached to the right hand it then had a scene component attached to it with its position offset to that it was in front of the muzzle of the gun. Then pushing the down button on the right trackpad would fire a projectile from the gun using the scene component as the spawning position. The projectile itself was

a sphere mesh with a projectile movement component set with an initial velocity so that it had a reasonable flight speed.

VR Threshold Pawn

The threshold pawn provides the input for answering the questions in the threshold experiment, in addition to some other minor features. Given that only two buttons were needed to answer the detection threshold question initially only the up and down buttons on the right trackpad were used as input to provide answers. However, after having performed the experiment a few times it became clear that the right HTC Vive controller that was being used while testing had lost some sensitivity on the trackpad. Because of this the participants would often have problems answering. As such the up and down buttons on the left controller were also mapped to be able to give answers. In the case that one of the buttons were pressed and the pawn was enabled the pawn would try to add the answer for the current gain to the results in the game instance object. In the case that up button was pressed 0 would be given as the answer while if the down button was pressed 1 would be given as the answer. If the required task had been completed and the answer was accepted an event was added to the tracking data specifying the choice and the gain. After that the array of remaining gains to test was checked to see if it was empty. This array was filled when the threshold pawn was create. It uses the range of gains and amount of iterations mentioned in the Method Chapter 3.3.2 to fill up the array which is then shuffled into a random order. To display the amount of gains remaining to the participants the length of the array was displayed on a text render component that was added in this class. In the case that the array was empty after an answer was given the "next instructions" function was called and the current gain was set to 0. In the case that the array was not empty the last index would be removed and the gain set to the new last index. Then the "next gain setup" function was called. The purpose of this function was for allowing resetting in the different sub-classes and customize depending on their needs. Last the text in the text render component was set to the new size of the array.

VR Threshold Rotation Pawn

This pawn implements the threshold experiment for rotation gain. To start with it creates two rotation targets. The rotation target object consists of a single sphere static mesh component. When this object was enabled and overlapped with the pawn it would tell the game instance object to accept threshold answers. In addition it will change color to blue from its default green color, disable itself and fire a triggered event. It was considered if the ball should change color to red as done in previous experiments[7], however this was judged to be an unimportant factor. To trigger the rotation targets a capsule collision component was attached to the

camera component of the pawn with a length of 10 meters as the rotation targets were set to always be 8m away from the camera. This capsule ensures that the overlapping event was triggered when looking at the rotation targets. The events triggered in the rotation targets was bound to the rotation pawn. So when one of these events triggered the pawn would disable the other target and adds a triggered event to the tracking data. The position of the rotation targets were update when the experiment started and whenever an answer was given. The function used to set the targets new position would always set the targets at a 90°rotation from the direction the user was looking. This was done by first getting the right vector of the camera component then multiplying it by how far away it should be from the camera. The final position of the two targets were calculated by adding and subtracting the multiplied right vector with the cameras position. The rotation targets were then set to their new position before being re-enabled making them green again. To ensure that the function was called when an answer was given the "next gain setup" function was overridden and made to call the function for setting the targets new position. The tick function was also used to keep the rotation targets the same height as the pawns camera component by setting their z position to be the same. Last it also implemented rotation gain as mentioned in section 4.6.1.

VR Threshold Movement Pawn

This pawn was used as a base for implementing the resetting of the curvature and translation threshold pawn as the resetting method was the same. Similar to the rotation threshold pawn this pawn also uses a target to evaluate was the participant has completed the given task before allowing them to answer. The movement target used in this case was almost exactly the same as the rotation target. The main difference was that it contains a scene component as the root with a box collision and a sphere static mesh attached. The box collision was in this case used to overlap with the pawn instead of a collision component on the pawn itself. The sphere mesh in the movement target was also placed 6m back along its relative x-axis, otherwise, it was the same as the one used in the rotation target. Last instead of setting the height of the object when matching the sphere height to the camera height only the sphere static mesh component was moved not the entire object. Instead of creating two targets at the start only a single movement target was created with this pawn. Its position was then set to be the pawns starting position plus the virtual walking distance required in the experiment along the initial transform's rotation's x-vector, the Z value of the x-vector was set to 0 to avoid the possibility of moving target up in the air. The movement target's rotation was also set to be the same as the pawn's initial rotation. By setting the start position marker on the floor used for the translation and curvature gain threshold experiment, as seen in the middle of Figure 7, so that start position marker's x-axis was aligned with the

path the movement target would be placed on the path and at the correct distance when the pawn was created. Though the rotation of the marker had to be in the opposite direction of the walking direction to make sure that the rotation of the movement target correct. In the case that it was not the ball static mesh would be between the users starting position and the box collision component. So with this, the movement target will be placed correctly within the virtual environment.

With this done the pawn needed to be rotated and position correctly. The first part was to rotate the actor so that the virtual path aligned with the diagonal of the physical room. Which was required to have enough space to perform the experiments. The aligning function was called when the menu button on the right controller was pressed. To align space the signed angle between the camera and the spawn location's rotation's x-vector was found, the z-values were set to 0 to only get the rotation in yaw. Then the same math was then done as in rotation gain 4.6.1 was applied except replacing the smoothed virtual rotation with the signed angle. The actors rotation was then recorded to avoid having repeat the alignment process, as the rotation would change when curvature gain was tested. However, before aligning the path was easier to first center the camera. Centering the camera entailed placing the camera component above the starting position by moving the actor. This was done by taking the spawn position minus the camera position flattening it by setting the z value to 0 and adding the resulting vector to the pawns world offset. This was required because when the translation and curvature experiments were done and the user had move back to the starting position in the real world it would no longer match the start position in the virtual world. Because of this the user would no longer be on the path or not have enough distance to move to the movement target. As such the moving the camera component to the start position was require after each test. In addition to avoid having the participant accidentally aligning the camera by accident the input was disabled when the begin button on the instructions were pressed.

Centering the camera was done when the side buttons on the right controller were pressed. In addition to centering the camera over the start position pressing the side, buttons would also set the actor rotation to the rotation that was found while aligning. The first time the side buttons were pressed a decal would be set at the current camera position to mark it. This decal would enable the participant to get back to the starting position without taking off the headset as it's relative location would always be over the same physical location. Otherwise, if the decal had already been set it would be hidden. In the case that the experiment had started pressing the side buttons would also make the movement target enabled and visible. Last the function would add an "in position" event to the tracking data.

Finally, the movement pawn also included an override of the "should apply

gain" and "next gain setup" function. The "should apply gain" function had a local Boolean that was used to determine if the participating had returned to the start position. The Boolean was additional AND parameter to the already existing enabled condition. So when it was set to false when the "next gain setup" function was called so no gain was applied when walking back to the start position. In addition, the function would disable and hide the movement pawn prevent it from being walked into again and to indicate to the participant that answer had been accepted. The function also set the reset position decal to visible. Then when the camera was centered again the Boolean would be set to true as long as the experiment had started, thus enabling the gain to be applied.

For the actual threshold translation pawn, all that needed to be done was to change the variables in the constructor to fit those mentioned in the Method Chapter 3.3.2 and add the correct instructions. The threshold curvature pawn in addition to the variables and instructions it also needed a few adjustments in order to facilitate the move 1.5 meters before applying gain rule. To do this the "should apply gain function" was overridden and another Boolean AND parameter was added to the conditions. This Boolean was used to signify that the participant had moved the required 1.5m. To do this in the pawns tick function if the participant had reset not reset the current camera position was stored in a previous position variable. In case that participant had reset the distance moved in x and y from the previous position to the current position would be calculated and added to a total distance moved variable. Then the current position would be set as the previous position. If the total distance moved exceeded 1.5m the Boolean would be set to true and the gain would be applied. The total distance moved would then be reset when the "next gain setup" function was called in the threshold curvature pawn.

4.6.1 Gain Implementation

The implementation of redirected walking for this paper took inspiration from the redirected walking toolkit [17]. However, given the different game engines used the final product became quite different. Given that the movement was always applied to the camera and controller components a rotation and translation gain value of 1 was in theory always applied to the relative components when working with redirected walking in the unreal engine. Because of this, the implementation of rotation and translation gain used a gain value 0 to indicate that no gain was being applied. As such applying a gain value of -0.1 to the pawn would be equivalent to applying a gain value of 0.9 overall. As such the gains for rotation and translation used in the implementation are the same as those mentioned in the Method Chapter 3.3.2 but -1. To make the gain values easier to compare between this thesis and previous studies the gain values used in the implementation are

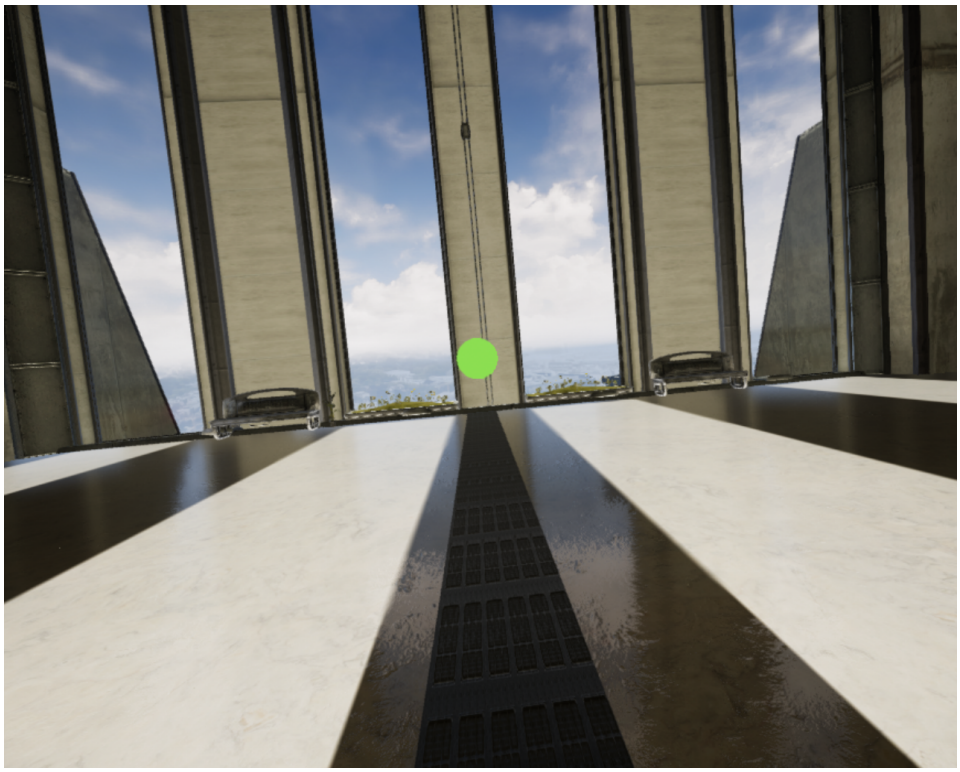

Figure 11: Ball to look at or walk towards during experiment

only mentioned here. As such the gain values are converted to the same range as used in the Method chapter and previous studies [32, 7] by adding 1 before being added to the results. With curvature gain none of the value mentioned in the Method Chapter are used when calculating curvature gain. Instead the circumference of the circle that would be made by the radius in $1/r$ was used to set the amount of curvature gain. Though given that the correct circumferences were used the effect will be the same. Again as with translation and rotation gain, the units given outside the Implementation will be in curvature gain or radius. The reason that it was implemented like this was that the code tried by looking at the redirected walking toolkit did not work. After having done the calculations on paper the solution the Author came up with was the one implemented. However, when calculating the curvature gain $1/r$ was thought of as the main focus was getting the angle to change by the correct amount based on the distance traveled. In order to avoid dividing by 0 or any other errors related to 0 values if statements where used to checked that the gain, delta movement or delta rotation was not 0 before calculated and applying gains.

Rotation gain

The initial step for applying rotation gain was calculating how much the pawn needs to be rotated ($R_{virtual}$) given the real rotation (R_{real}) and the rotation gain applied (gR).

$$R_{virtual} = R_{real} * gR$$

For rotation gain, it was required to smooth the rotation as low rotation gain values caused stuttering. To calculate the smoothed rotation ($sR_{virtual}$) the last rotation ($lR_{virtual}$) applied was required along with a smoothing factor (sF) of 0.5.

$$sR_{virtual} = (R_{virtual} * sF) + (lR_{virtual} * (1 - sF))$$

Having found the rotation the next step is to find the vector (Vec) for how much the pawn needs to be moved so that the rotation applied to the pawn will cause the camera component to be rotated while staying in the same position. To find this position we first get the vector (V_{CP}) from the camera component (C) to the pawn (P).

$$V_{CP} = O_{pos} - P_{pos}$$

Then the vector is rotated around the z-axis. We then subtract the rotated vector by the vector from the camera component to the pawn root again as this will give us the vector for how much the actor is required to move, as the rotated vector only gives how much the camera component would need to move to reach the same position.

$$Vec = RotateAroundAxis(V_{CP}, sR_{virtual}, Z_{axis}) - V_{CP}$$

The Vector (*Vec*) and smoothed virtual rotation ($sR_{virtual}$) are then added to the pawns world transform.

Translation gain

Before applying the translation gain (gT) the changes in real movement (T_{real}) needed to be rotated to match the pawns rotation. If not the translation gain would not be applied in the right direction.

$$rT_{real} = RotateAroundAxis(T_{real}, P_{rot.z}, Z_{axis})$$

Then how much the object should be moved can be calculated.

$$T_{virtual} = Normalize(rT_{real}) * (|rT_{real}| * G_t)$$

The virtual translation ($T_{virtual}$) is then added to the pawn world offset.

Curvature gain

Curvature gain (gC) is implemented similarly to rotation gain with the exception being how the virtual rotation ($R_{virtual}$) to apply is calculated.

$$R_{virtual} = (\frac{|T_{real}|}{Ci}) * 360$$

Then as with rotation gain, the same math is used to calculate what position the pawn needs to be moved to for the rotation to only apply to the camera, but without the smoothing. The Vector (*Vec*) and virtual rotation ($sR_{virtual}$) are then added to the pawns world transform.

5 Results

This chapter contains the main results from the two experiments described in the method Chapter 3. Each of the results from the experiments has been separated by the gain being tested. As mentioned in the Implementation and Method Chapter eye height, gait and movement tracking data were also gathered. However, analyzing this data was beyond the scope and time of this thesis. The raw data can be found in Appendix D together with additional data that did not fit in the Results Chapter.

5.1 Rotation Gain

When testing rotation gain a total of 17 participants were selected through convenience sampling. On average each test needed 20 to 40 minutes to complete. Out of the participants 14 identified as male and 3 as female. The participants were all between 20-30 years of age. On average the participants had large amounts of gaming and VR experience. 7 of the participants used some sort of vision correction and 4 used their vision correction while wearing the HMD. The individual answers can be seen in Table 18. The calculated simulation sickness scores can be seen in Table 21. On average the Pre-SSQ score was 10.52 with a Post-SSQ score of 27.82.

5.1.1 Increment Experiment Rotation

The graphs in this subsection show the rotation gain values at which participants indicated that they noticed the rotation gain being applied and when the gain became unusable for both positive (Figure 12) and negative (Figute 13) rotation gain. The individual values can be seen in Table 12 in the appendix. The results of one participant were removed due to their miss understanding of the task. The average detection point for positive rotation gain was at 1.835 rotation gain and the average discomfort limit was at 2.589 rotation gain. The ratio between noticeable and discomfort limit was 1.0 : 1.41 for positive rotation gain. With negative rotation gain, the average detection point was at 0.69 rotation gain and the discomfort limit at 0.555 rotation gain. The ratio for noticeable rotation gain to discomforting rotation gain being 1.0 : 0.8.

5.1.2 Detection Threshold Experiment Rotation

During this experiment four (2 female, 2 male) of the participants were unable to complete the task because of simulation sickness and one female participant was

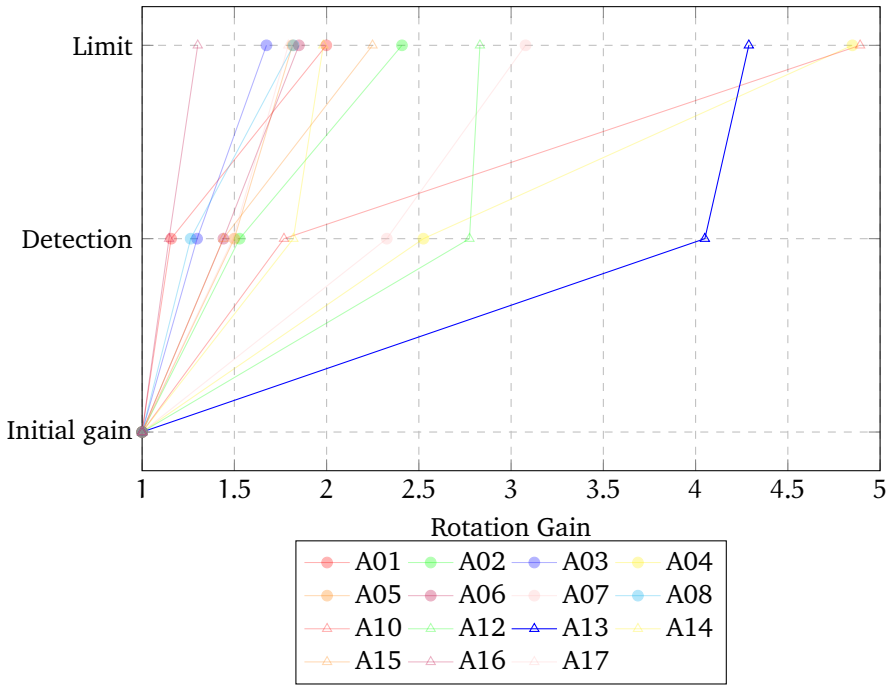

Figure 12: Individual positive rotation gain values from start of experiment to noticeable and then to their limit

unable to complete due to technical issues. As such their data was not included when calculating the average probability of virtual rotation being less than real rotation response, as seen in Table 15. The resulting graph from fitting the probabilities with the following psychometric (sigmoid) function [7]

$$f(x) = \frac{1}{1 + e^{a*x+b}}$$

can be seen in Figure 15. Based on the results of the graph in Figure 15 a point of subjective equality (PSE) (50%) of 0.96 rotation gain was found, along with an upper detectable threshold (75%) of 0.79 rotation gain and a lower detectable threshold (25%) of 1.13 rotation gain. In other words, the graph shows that the detection threshold for rotation gain is at +13% rotation and -21% rotation. The detection threshold for each participant was also calculated and can be found in Table 11. The standard deviation for rotation gain was 0.360 rotation gain for positive gain and 0.143 rotation gain for negative gain. This data was then used to perform two-way student t-tests against the results from the studies with old hardware and similar conditions to find if the results were statistically different. The p-values of the t-test can be seen in Table 4 and Table 5. As the studies that

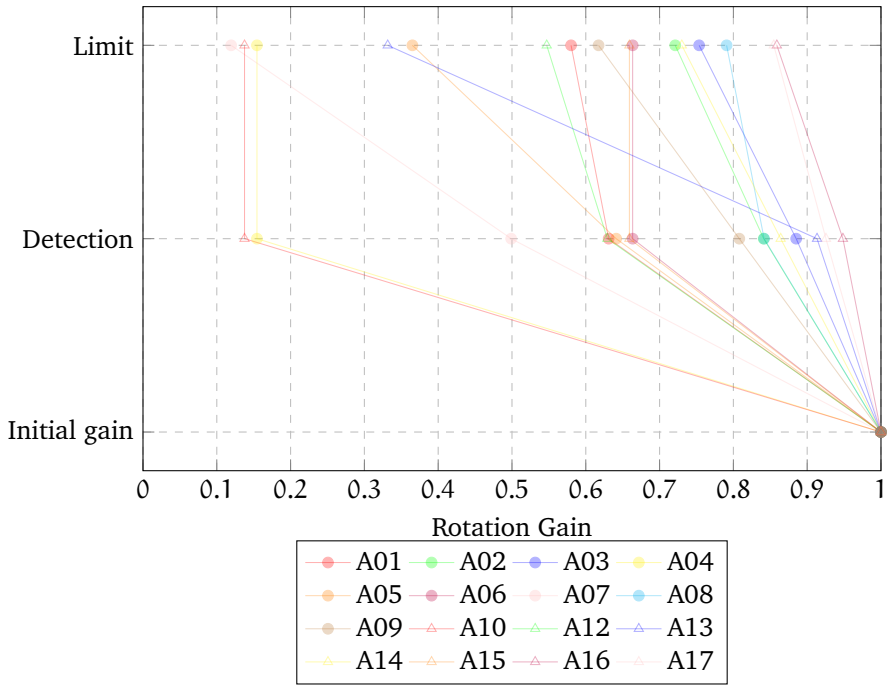

Figure 13: Individual negative rotation gain values from start of experiment to noticeable and then to their limit

were being compared to did not include the individual detection threshold it was not possible to know what their standard deviation or mean was. In the case of the mean, the detection thresholds could be used instead. The Author attempted to get information regarding standard deviation from the authors of those studies [32, 7, 21] that were being compared against, however that data was no longer available. As such two t-tests were done for when comparing, one which assumes that the standard deviation was the same and one that assumed that their standard deviation was twice of that in this study. As a critical value, 95% (0.05) was used.

Paper	p-value	p-value (std*2)
Stenrich 2010	1.0000	1.0000
Stenrich 2008	0.1008	0.2904
Bruder 2008 (Male)	0.5642	0.6870
Bruder 2008 (Female)	0.8905	0.9212

Table 4: P-values for negative rotation gain

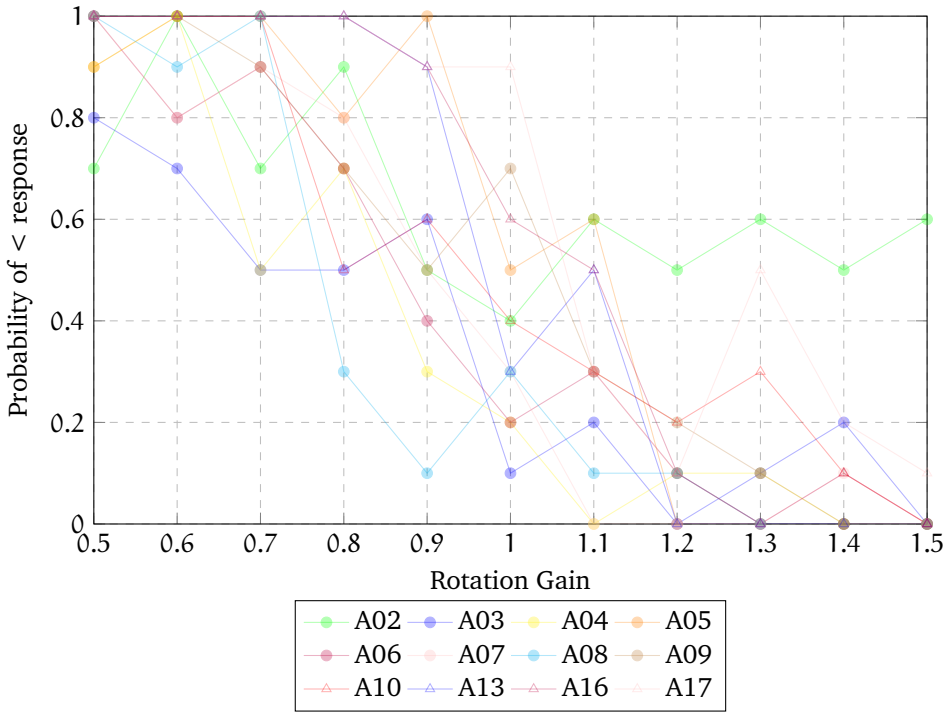

Figure 14: Probability of individual choosing that the virtual rotation < real rotation for each gain during experiment

5.2 Translation Gain

When testing translation gain a total of 11 people were recruited through convenience sampling, with all being able to complete both parts of the experiment. The time needed to complete the test was on average between 40 to 60 minutes. All participants identified as male and were between 20-30 years of age. On average the participants had large amounts of gaming and VR experience. 5 out of the 11 participants used some sort of vision correction and 3 used the vision correction while wearing the HMD. The individual answers of each participant can be found in the appendix in Table 19. On average the Pre-SSQ score was 4.49 with a Post-SSQ score of 6.73. The individual SSQ scores are in Table 22 in the appendix.

5.2.1 Increment Experiment Translation

Table 13 shows the translation gain values at which participants indicated that they noticed the translation gain being applied and when the gain became unusable for them for both positive and negative translation gain. The average detection point for positive translation gain was at 1.491 translation gain and the average discom-

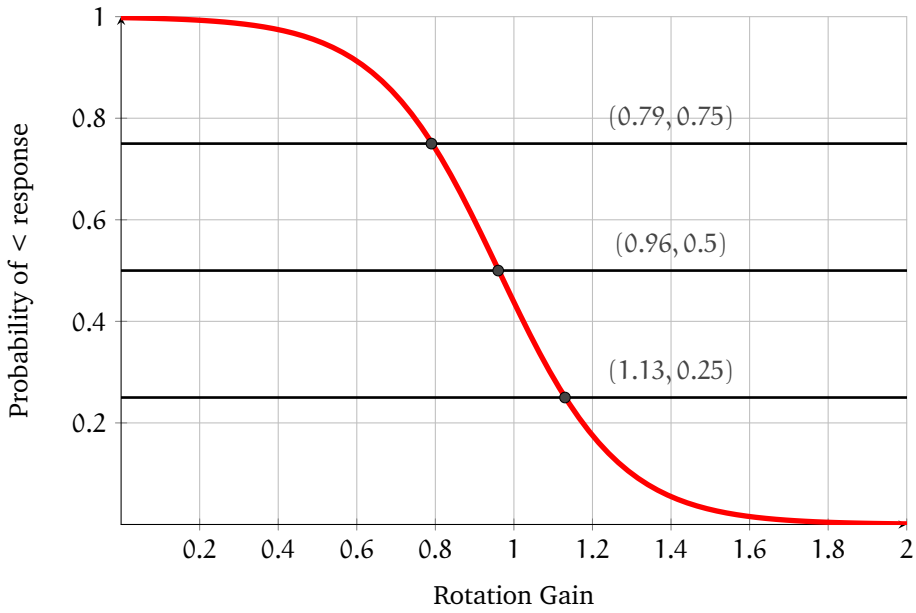

Figure 15: Psychometric function fitted to the average probability of virtual rotation < real rotation response from the users.

Paper	p-value	p-value (std*2)
Stenrich 2010	0.0371	0.1807
Stenrich 2008	0.0022	0.0396
Bruder (Male)	0.1346	0.2892
Bruder (Female)	0.078	0.1952

Table 5: P-values for positive rotation gain

fort limit was at 2.242 translation gain, making the ratio between noticeable and uncomfortable gain 1.0 : 1.504. With negative translation gain, the average detection point was at 0.742 translation gain, with a ratio of 1.0 : 0.71 for noticeable and uncomfortable translation gain.

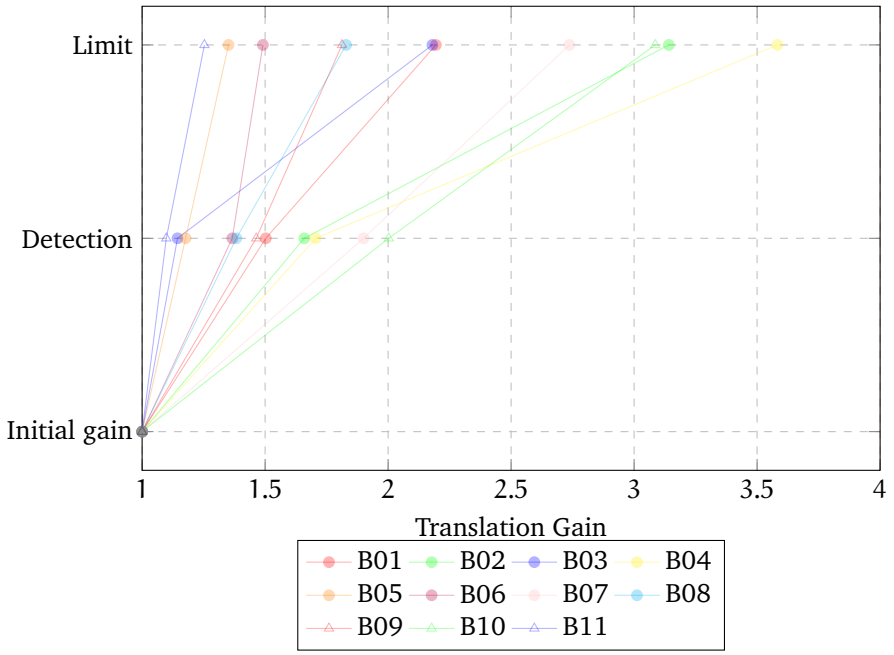

Figure 16: Individual positive translation gain values from start of experiment to noticeable and then to limit

5.2.2 Detection Threshold Experiment Translation

The results from fitting the same psychometric function as used in rotation gain to the data in Table 16 can be seen in figure 19. By analyzing the resulting graph a PSE (50%) value of 1.07 translation gain was found, with an upper detection threshold (75%) of 0.95 translation gain and a lower detection threshold (25%) of 1.19 translation gain. This suggests that virtual movement can be scaled down by -5% and scaled up by +19%. When preparing to perform the student t-test the standard deviation for increased translation gain was 0.165 and for reduced translation gain was 0.096. The individual detection thresholds for translation gain can be found in appendix Table 10. The p-values from the student t-tests are in Table 6 and Table 7. The same conditions were applied as when calculating for rotation gain.

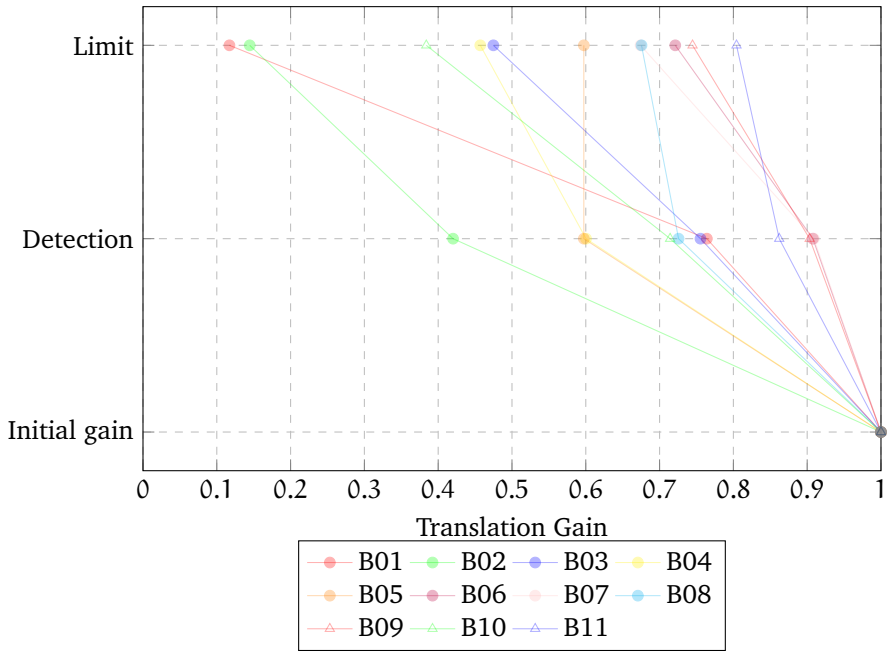

Figure 17: Individual negative translation gain values from start of experiment to noticeable and then to limit

5.3 Curvature Gain

When testing curvature gain a total of 14 participants participated in the testing, and each test took about 30 to 40 minutes. All participants identified as male and were between 18-30 years of age. On average the participants had large amounts of gaming and VR experience. 6 out of the 14 participants used some sort of vision correction and 4 used the vision correction while wearing the HMD. The individual answers of each participant can be seen in Table 20 in the appendix. On average the Pre-SSQ score was 9.88 with a Post-SSQ score of 14.96. The individual SSQ score can be seen in Table23 in the appendix.

Paper	p-value	p-value (std*2)
Stenrich 2010	0.0042	0.0618
Stenrich 2008	0.0144	0.1205
Bruder (Male)	0.0592	0.1827
Bruder (Female)	0.0934	0.2246

Table 6: P-values for negative translation gain

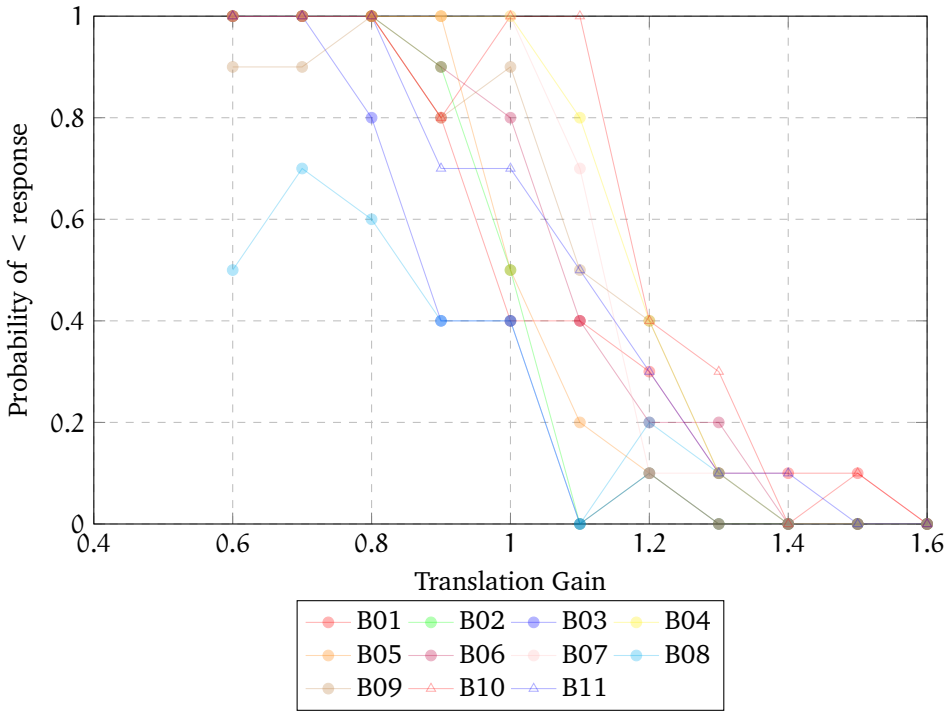

Figure 18: Probability of individual choosing that the virtual movement < real movement for each gain during experiment

5.3.1 Increment Experiment Curvature

Table 14 shows which curvature gain values the participants noticed the curvature gain was being applied and at which curvature gain they became uncomfortable. As either positive or negative curvature gain was done randomly the values were merged into one Table 14, as seen in the appendix. Two of the participant's data was removed as they miss understood the task. The average detection point for combined curvature gains was 0.20 (8.42m radius) curvature gain and the average discomfort limit was at 0.28 (4.96m radius) curvature gain, making the ratio for noticeable to uncomfortable translation gain 1.0 : 1.4.

5.3.2 Detection Threshold Experiment Curvature

As the method used for finding the detection threshold for curvature gain only used one rotation no PSE is measurable. The data from Table 17 was fitted to the same psychometric function as used with translation and rotation gain. The detection threshold (50%) found based on the fitted function as seen in graph 22 was at 0.045 curvature gain. Converted to a radius the detection threshold would be

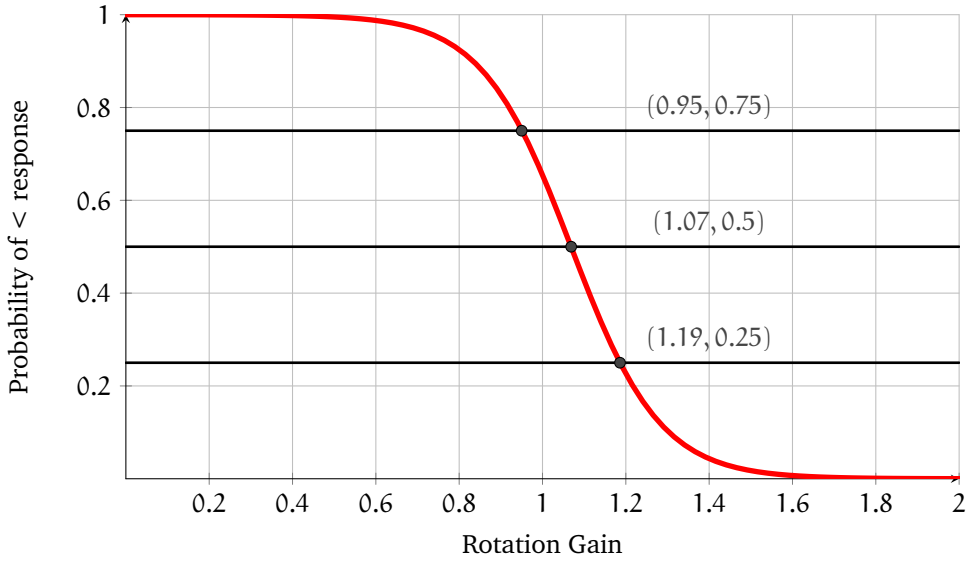

Figure 19: Psychometric function fitted to the average probability of virtual movement < real movement response from the users.

Paper	p-value	p-value (std*2)
Stenrich 2010	0.8017	0.8765
Stenrich 2008	0.3118	0.5332
Bruder (Male)	0.425	0.5808
Bruder (Female)	0.8088	0.8636

Table 7: P-values for positive translation gain

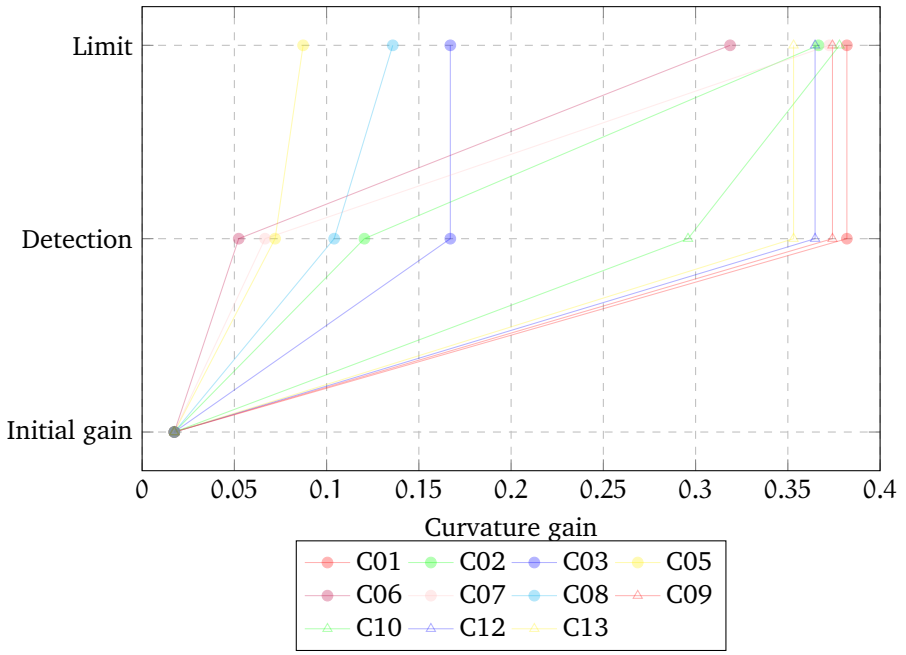

Figure 20: Individual curvature gain values from start of experiment to noticeable and then to limit, negatives made positive to fit same graph

Paper	p-value	p-value (std*2)
Stenrich 2010	0.9655	0.9757
Stenrich 2008	0.7658	0.8437
Bruder (Male)	0.3187	0.4777
Bruder (Female)	0.713	0.7884

Table 8: P-values for curvature gain

22.22m. The detection threshold each participant was also calculated and can be found in the appendix in Table 11. The standard deviation found was 0.026 curvature gain and the p-values of when compared to previous studies are in Table 8. The individual thresholds can be seen in Table 11 in the appendix.

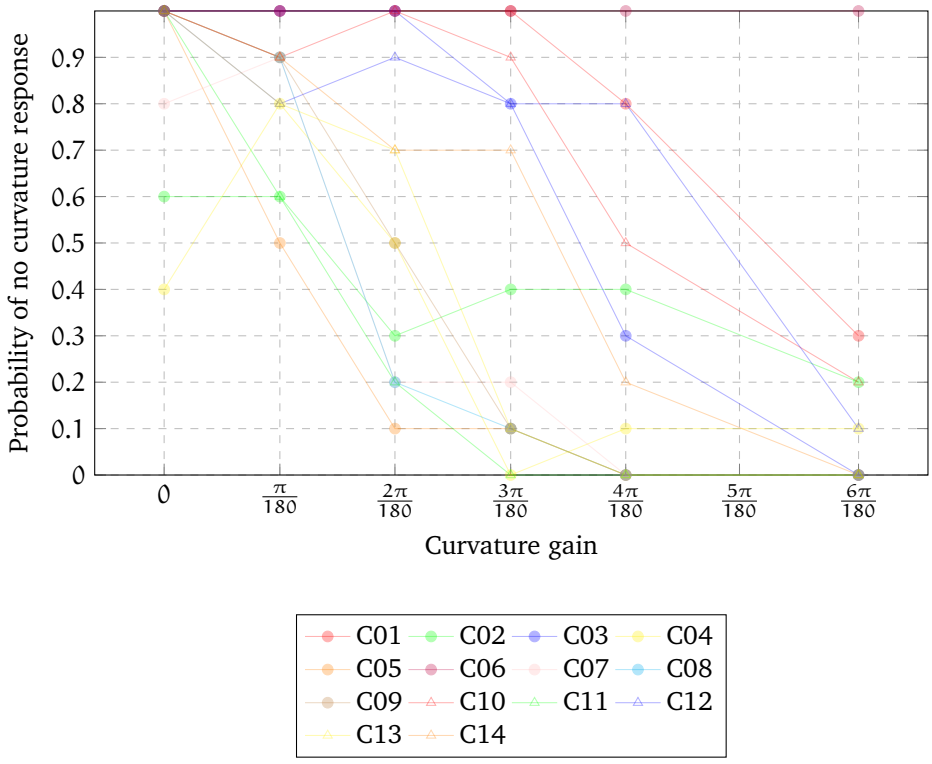

Figure 21: Probability of individual choosing that the no curving for each gain during experiment

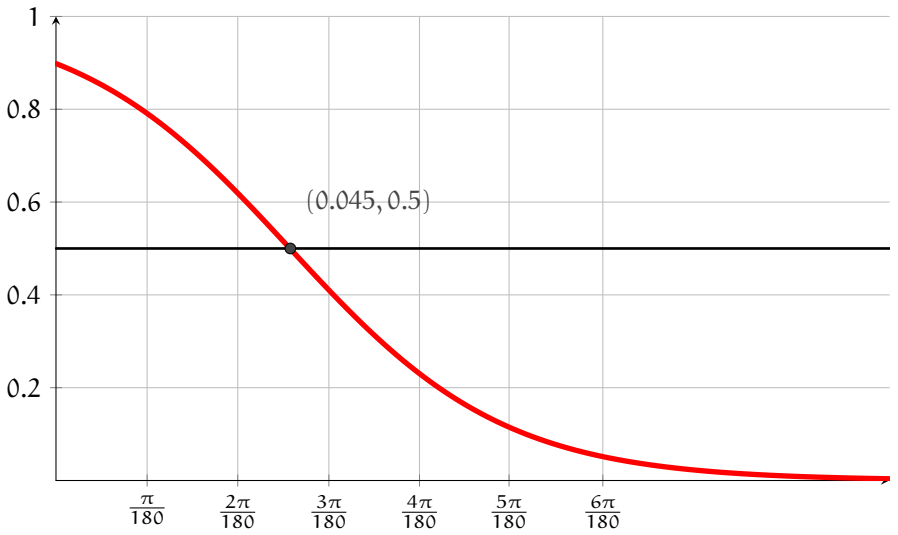

Figure 22: Psychometric function fitted to the average probability of no curvature response from the users.

6 Discussion

This chapter contains a discussion of the results found. For the observed changes in detection threshold factors other than hardware are discussed. In addition to which aspect of the hardware caused the observed changes and what are the consequences. For the increment experiment outliers in the data are discussed, as well as how the results could be used as a threshold for steering algorithms.

6.1 Detection Thresholds

Based on the results of rotation gain the calculated detection thresholds shows that the rotation can be increased by 13% and decreased by 21%. When comparing these results to the previous studies of Steinicke et al. [7, 32] and Bruder et al. [21] the biggest change is how much the rotation can be increased. With the difference ranging from about 31% - 55%, while the amount the rotation can be decreased only varies by 1% - 10%. In the point of subjective equality (PSE) is about the same as in this thesis as those in Steinicke et al. [7] found in 2010 and Bruder et al. [21] study from 2009. The difference in PSE with Steinicke et al. [32] study from 2008 comes from the questions used in that study as they themselves point out. As such, there is not a great difference in bias. A similar trend can also be seen in other studies with improved hardware such as Bruder et al. [35] in their 2012 study, Paludan et al. [33] and Chen et al [37]. As their detection threshold for increased rotation gain is also lower compared to the studies mentioned before. While the detection threshold for how much the rotation gain can be decreased is still more or less within the 10% to 20% range. Though if this trend is caused by the change in hardware in these studies [35, 33, 37] is difficult to say given the low amount of iterations used, as mentioned in the Related Work Section 2.1. So even if those studies show a similar result it is nothing conclusive. Having performed the student T-test's to calculate if there is a statistically significant difference in detection thresholds. The results show that there is no statistically significant difference for decreased rotation gain. On the other hand for increased rotation gain the student T-test's show that there is a significant difference between the results in this thesis and those from the studies [7, 32, 21] were the FOV was 40. In the case of Steinicke et al. [32] study from 2008 the difference is statistically significant even with double standard deviation. Why the results are not statistically significant to those of Bruder et al. [21] study from 2009 is most likely because of the low number of participants in that study. As such there is some statistical evidence

for the change in the detection threshold for increased rotation gain is not caused by sampling error.

Regarding the individual results for rotation gain found in Graph 14 participant A2 is considered an outlier in the data. As the data suggests that they are not able to distinguish increased rotation gain from decreased rotation gain when increased rotation gain was applied. However, the participant was not removed from the data set as no valid reason for doing so was found. Given that there was only one outlier it is unlikely that it had a large effect on the detection threshold. Another factor that might have influenced the detection threshold for rotation gain would be the implementation of rotation gain. Specifically for rotation gain smoothing was added to the rotation to avoid camera stuttering when rotation gain was decreased. The cause of the stuttering was most likely caused by interactions with the game engine as previous studies do not mention requiring smoothing. As such removing the stuttering would return the rotation gain to its normal state rather than stuttering being a feature of rotation gain. Though to ensure that the rotation was not significantly affected a 360° virtual rotation was performed with 1.5 rotation gain while both the smoothed and un-smoothed rotation was recorded. Based on the results from the collected data, as seen in Table 28, the rotation was only changed minutely with each calculation. The amount was so small that the participants should not be able to notice a difference. As such, it is reasonable to expect that it did not have an effect on the detection threshold. Another aspect of the implementation of rotation gain that might have had an effect on the measured detection threshold is the fact that no consideration to cable management was taken in this implementation. In the studies performed by Steinicke et al. [32, 7] and Bruder et al. [21] a reset was performed between each rotation so that the cable would not be tangled. In this experiment, no such measures were taken, as the participants could simply alternate between left and right rotation to avoid the issue. However, with the reset, the rotation gain would always change from no rotation gain to the rotation gain being tested. As such the rotation gain would at the most go from 1 to 1.5 rotation gain or 1. to 0.5 rotation gain. Whereas in this implementation the change could go from 0.5 rotation gain to 1.5 rotation gain or opposite depending on the order the gains were being tested. However, on the other hand, it also had the possibility of going from 1.4 rotation gain to 1.5 rotation gain, which was not possible in previous studies. As such the difference between each gain tested was not as consistent in this implementation. The reason it might matter was if the participants used the difference in rotation between each gain when answering instead of doing a proper evaluation of the rotation. Though on average it might not make a difference. In the case, this did influencing the detection threshold the expected effect would be that both positive and negative detection thresholds would change.

As the increases and decreases in gain between test are just as frequent. As the results only indicate a significant change in the positive detection threshold, not the negative this would suggest that it is not a major factor in causing the change in the positive detection threshold.

For translation gain, the results found that the detection thresholds were at 19% increased and 5% decreased translation gain. When comparing this results to the previous studies of Steinicke et al. [7, 32] and Bruder et al. [21] there is a small difference in the detection threshold for how much the translation gain can be increased. However, the detection threshold for decreased translation gain stands the most out. As the previous studies all having found a detection threshold for decreased translation gain at around 20%. Again as with rotation gain, the PSE value found almost the same with the exception of Steinicke et al.'s [32] study from 2008. The reasons being the same as mentioned before. The results from the student T-test further suggest this to be the case as no significant difference in detection threshold for increased rotation gain, but a statistical difference is found for the detection threshold for decreased translation gain. Specifically when comparing Steinicke et al. [7, 32] and assuming the same standard deviation.

Regarding the method used in this thesis when testing the translation gain. Two additional increased translation gains were added to the range of gains tested compared to previous studies. As such 4 decreased gains were tested while 6 increased gains were tested. It is possible that the imbalance between the positive and negative gains tested might have had an effect on the detection threshold. A possibility is that it made the reduced translation gains more noticeable in comparison to the more numerous increased translation gains. Thought this is similar to the issue mentioned in rotation gain where the difference between the gains being tested could cause judgment bias. Though in this case there is a reset between each gain tested. As such it is less likely that the additional increased gains affect the answer given for the next gain. In addition, when calculating the results without the last two increased translation gains there was no change to the detection threshold or the individual detection thresholds. As all most all of the participants were able to notice correctly when those gains were being applied.

With curvature gain, the detection threshold found would cause the participants to walk in a circle with a radius of 22.22m. Which was not that different from the results of Steinicke et al. [32] second experiment in their 2008 study, Steinicke et al's [7] study from 2010 and Bruder et al. [21]. In addition, when performing the student T-test comparison no statistically significant difference was found. As such, these results suggest that hardware changes have no effect on curvature gain.

In regards to the outliers in the data, the results from participant C6 is the most apparent. As the participant was not able to tell that curvature gain was

being applied at all. As such it was considered if the results should be removed. However, other participants such as C1 also didn't notice the curvature until the very end. As such the curvature gains tested were probably to low for participant C6 to notice. Having no other reasons to dismiss the results they were kept in. In regards to the method used in the test environment. One was changing the distance without curvature gain from 7m as in previous studies [7, 7, 21] to 6.5m to accommodate the room space. Though a slight difference it is unlikely that this would have caused many variations, as it is still a reasonable length to move before the gain kicks in. Another variation from previous studies is how the reset was done. In earlier studies, the scene was changed to a blank scene with two markers when re-positioning between tests [32, 7]. In this implementation, the virtual scene was not changed and the marker was added as is. This meant that the users could, based on the marker's offset to the path, evaluate how much curvature had been applied. However, as this was only visible after the answer was given it should not have influenced the results.

In addition to the factors that only affected the individual gain, there were other factors that could influence the detection threshold for all gains. One such factor is the virtual environment used in the experiments. As the virtual environment in this thesis is has a skyscraper aesthetic while the previous studies [21, 32, 7] used city aesthetics with varying quality and visual density. In regards to visual density, the results suggest that an adequate amount of visual density is present in the virtual environment used. If there was not enough visual density the answers in the graphs from Figures 14 18 21 would all be around 0.5 probability for all gains. As participants would not be able to tell if a gain was applied and would have to guess, as seen in Paludan et al's. [33] study on the effect on visual density. Regarding the quality of the virtual environment effects on the detection threshold. Previous research [22] has shown that the texture and illumination at least do not affect human perception when using redirected walking. As such, it is suspected that whether the assets are high quality or low makes no difference, as long as the environment provides the visual density necessary for the participants to notice a gain being applied. The same arguments are also applied to the setting of the virtual environment. Given that the implementation in this study has similar placement constrictions as previous studies it is unlikely that the changes to aesthetics would have a significant impact on the detection thresholds.

The implementation of the gains has also changed a deal from how it was done in previous studies [32, 7] as mentioned in the implementation Chapter 4. A specific part that might have affected the detection threshold is the how the change in physical movement was calculated. As the implementation does not distinguish between the forward and right vector when applying gains. As such translation

and curvature gain are also applied to sideways movement. Though while this is the case, the tasks the participants performed during translation and curvature experiments were explicitly performed using forward walking. At most translation gain could be expected to increase simulation sickness because slightly increase swaying motions. However, based on the SSQ results translation gains experience the least change in score suggesting that the predicted effect did not happen. Probably because the gains used when testing was not large enough for this effect to be noticeable. In addition, while the implementation differs the applied gain still produce the same effect as in previous studies [32, 7]. As such the implementation of the gains should not affect the detection thresholds.

Last regarding how the pseudo-2AFC questions where formulated. For rotation and translation gain, the questions are initially the same as those used by Steinicke et al. [7] in 2010 and Bruder et al. [21] in 2009. With rotation gain asking the participants to evaluate if the virtual rotation was lesser or greater than the physical rotation and for translation gain asking if the virtual distance moved is lesser or greater than the physical movement. The difference in this thesis was that an alternative question was provided. In the case of rotation gain, the alternative question the participants were asked to evaluate was if the rotation speed was slower or faster than normal rotation. For translation gain, the alternative question asked was if the movement speed was slower or faster than normal movement speed. The first reason this was done was to make it easier for the participant to give their evaluation. As having to remember to press upon the controller when the virtual rotation was greater than the physical rotation is significantly harder than remembering to press up when the rotation was faster than normal. The second reason is that it would become apparent to the participants that when the rotation in the virtual world is less than the physical world the rotation will feel slow and fast when the virtual rotation is larger. The same principle applying to translation gain. As such the instructions only point out an observation the participants would notice by themselves after a few tests. As such adding those instructions would not affect the detection threshold.

Other than the implementation of the experiments the participants themselves might have had an effect on the detection threshold. First, in regards to the number of participants tested in this thesis, the amount is this thesis is similar to those of other studies calculating detection thresholds. As the other studies mentioned in the related work Section 2.1 also had a similar number of participants ranging from 10 - 15. A more challenging feature of the demographics for this thesis is the gender distribution. With all participants being male for the detection threshold experiment, it is likely that there is bias in the results related to gender and visual processing. The results from Bruder et al. [21] suggests there is a differ-

ence between males and females when evaluating detection thresholds. However, because there were few participants in that study, the significance of the result is questionable. Though their results indicate that males have slightly lower detection thresholds, but larger PSE values than females. As such if the experiment had more gender balance slightly larger detection thresholds and lower PSE could be expected. Though slightly larger detection thresholds would only mean an increase by between 2.5% to 4%, and possibly an increase of 4m on the radius for curvature, based on the differences seen in Bruder et al. [21] study. Such a small variation would likely not change the result found when performing the student T-Tests. In addition, some of the studies [21, 32] the results were compared to also feature a low number of female participants. As such the lack of female participants certainly had some effect on the detection threshold, but not enough to explain the results found. Additionally, most of the participants in this experiment had some experience with HMD's. If this gave any of the participants any advantages to detecting the thresholds is unknown. The author speculates that it would only improve the amount of simulation sickness they experience, with those that have used HMD's a lot experiencing less do to acclimation effects. The participants in this experiment were also between the age of 21 - 25, however, the effects of age on detection thresholds is also not known. The same is true of using vision correction tools such as glasses or contact lenses.

Based on the results found through the detection threshold experiment and with the student T-test comparison with earlier measurements. The results indicate that the detection threshold has been significantly changed for increased rotation gain and decreased translation gain. As discussed there are few factors other than the specifications of the hardware itself that are contributors to the observed changes. The most likely possibilities other than hardware is that the difference was caused by because the sampled participants were more sensitive to these gains. Given that the number of participants was only between 10-14 for each of rotation, translation and curvature gain. As such a larger amount of participants would show if this was the case. However, due to the time requirements of this type of experiment and the time limitations on this thesis more experiments were not performed. On the other hand, each of the participants evaluated each of the gains in the tested range 10 times each which makes the results from this thesis have high accuracy in relation to other studies within the field of detection thresholds in redirected walking. Though the results were compared with studies [32, 7, 21] with as similar as possible implementation, however, because no standard deviation could be obtained from those studies it does question how reliable the student T-test results are. While the p-values are below the critical value when comparing against Steinicke et al.'s [32, 7] studies if the same standard deviation is assumed. How-

ever, if the standard deviation is doubled only one p-value for increased rotation gain is below the critical value, while one p-value for decreased translation gain is very close to being below the critical value. As such having had more participants and possibly more iteration would definitely improve the confidence in the standard deviation. That said the Author still believes it is reasonable to reject the null hypothesis given the evidence presented. As such, the hardware changes have had an effect on the detection threshold in redirected walking.

In regard to what specific aspect of the hardware that is causing the observed difference in detection threshold, the author believes the change in the field of view (FOV) to be the most likely candidate. As the changes in resolution is required with a higher FOV to maintain the pixel per degree ratio. In the case of HMD used by Steinicke et al. [32, 7] the resolution was 800x600 per eye with a 40°FOV making the ratio 20 pixels per degree. In this thesis, the HTC Vive used has a resolution of 1080×1200 per eye with a FOV of 110°making the ratio about 10 pixels per degree. As such, the concentration of pixels would have been sharper in the previous studies and would create a clearer image. Though even with a lower pixel per degree ration the rendered image on the latest hardware is still very clear, as seen in Figure 11. As none of the HMD discussed should provide a visual scene where the objects are too pixelated to be used to judge distance or acceleration it seems unlikely that the change in resolution is the cause of the observed results. Regarding frame rate, the reason that the latest HMD target 90Hz is to avoid simulator sickness [41]. As lower frame rates can cause motion blurring. As such their might have been more motion blur in earlier studies were the frame rate was at 60Hz. In addition, the higher frame rate might have allowed for a more accurate sense of motion. A more accurate sense of motion could make it easier for the participants to detect when a gain was applied. Which would then lower the detection thresholds, as seen in the results. As such, frame rate might be the cause of the results observed. On the other hand, the change in FOV from 40 to 110 would dramatically increase how much of the scene is visible at any given time. It would also give the participants much more peripheral vision. Peripheral vision has been shown to be a factor in speed perception in non-VR virtual environments [43]. In addition findings by Nilsson et al. [44] suggest that the smaller the FOV the more users would underestimate the virtual walking speed when using walk-in-place locomotion. A similar effect may be taking placing with translation gain as well. As the increased FOV allows for a better evaluation of virtual walking speed makes it easier for the participants to notice that the gain is being applied. As to exactly why the differences in HMD causes specifically the detection thresholds of increased rotation gain and decreased translation gain to be worse is unknown.

With the lower detection threshold found in this thesis steering algorithms [6,

18, 19] will become less efficient. As they can no longer apply the same amount of redirection as before. This would again lead to more resets [8] when using redirected walking and which leads to a less immersed experience. As such steps are needed to increase the detection threshold on equipment with higher specifications. One possibility would be to instigate low FOV specifically when the affected gain is applied. This could mean blacking out the sides of the screen similar to what is done when using fly mode in Google Earth VR. Though this might be more very noticeable to the user. Another possibility might be to combine it with eye tracking to optimize the effect[28].

6.2 Usable and Noticeable Gains

Based on the results from the increment experiment many of the participants continued the experiment even though the gain became noticeable. This indicates that when the gain becomes noticeable is not when the gain becomes uncomfortable. Though how large the range between noticeable and uncomfortable is clearly distributed between individuals. With a few instances of participants become uncomfortable the moment the gain became noticeable while a hand full are able to handle the applied gains up to the implemented limit. A special case of this can be seen in regards to the results from curvature gain, as a large number of the participants reached the limit without noticing the gain being applied. In contrast when the detection threshold experiment for curvature gain was performed most of the participants were able to notice that the curvature gain was applied compared to this experiment. The main differences between these two tests was the environment and the task performed. As in the detection threshold experiment the participants were required to walk along the line on the floor. The Author believes that the floor line to be the greatest visual difference between the two experiments as both rooms otherwise uses the same floor and have a sufficient amount of visual density in regards to vertical objects. As such, the Author suspect that one reason that many participants were not able to notice the gain was because of a lack of visual density needed to specifically notice curvature gain. Further evidence of this comes from observation of the participants that noted during the experiment that they were able to see the effect of the gain on the boundary of the tracked space, but where unable to notice anything when only relying on the virtual environment. Another factor was also that the curvature gain in the detection threshold experiment would go from no gain to the tested gain after 1.5m of walking, creating a markedly more sudden shift as compared to the slight increases in the increment experiment. The same effect was observed by Steinicke et al. [32] as their initial curvature gain test with 2m of no gain made their participants far more uncertain regarding if gain was applied. A possible solution to have made the curvature gain

easier to detect would be to add linear lines to the floor that can be used to judge if the gain is applied. Slightly similar with rotation gain certain individuals did not notice the rotation gain being applied at all or only very late. In the case of the negative rotation gain instances where the limit was hit, it is very unlikely that the participants did not notice the gain. It is more likely that they did not press the button correctly or misunderstood the task. However since that could not be proven they were not removed. Even with this being the case, the other test instances still indicate that the gain did not become uncomfortable upon being noticeable. As such this goes against the null hypothesis that states that the participants would become uncomfortable with the gain as soon as it was noticed, as such the null hypothesis will be rejected.

Using the difference between noticeable and uncomfortable gain from this thesis to increase the gain limits for redirected walking algorithms is debatable. Given that the results produced in this experiment do not represent a worst-case scenario, such as the results from detection threshold. This is because several factors contribute to making the applied gain less noticeable. Such as the virtual environment which provides distractions for the user. Further, as the participants were told to do what they wished within the area they did not always perform the movements that were optimal for detection the gain at all times. As such, the participant might stand still for a time or make small slow movements. In addition, the shape of the room made it difficult to walk distances over 5m without hitting the edge of tracked space. As such, using something like the average point where participants became uncomfortable as a maximum value for gains is not advised. A possible way to find the threshold for when the gain would be to perform the pseudo-2AFC experiment for finding the detection threshold, but with a larger range and different alternatives. Though given the nature of simulation sickness evaluating if a given gain causes unpleasantness might not be repeated multiple times, as simulation sickness tends to build up. Further, as the range that participants became uncomfortable is wider than detection based on the results found, using a large step between each gain tested would probably be required. Because of this, a method of constant stimuli is probably not the best choice. As an alternative, an adaptive method [10] could possibly be used instead, as it reduces the number of tests needed to be done.

Another possible way to use the results from this study would be to use the ratio between no gain to noticeable gain and noticeable gain to uncomfortable gain. This ratio could then be applied to the detection threshold as it would serve the same as the no gain to noticeable gain change in gain. The resulting gain could then be used as a possible discomfort threshold. Though to find this threshold there would need to be a larger number of more specific experiments.

In the case that a reasonable discomfort threshold is found, it would probably

be best for any implementation to provide the discomfort threshold as an option to the detection threshold and not a default. As the results of this thesis show that for some instances the experience becomes uncomfortable the moment the gain is noticeable. Special care is also required to as to which gains are made noticeable. As based on the SSQ score rotation gain is the gain that causes the most amount of simulation sickness. Within rotation gain, it is specifically negative rotation gain that is the problem, as observed during testing. This can also be seen in the low range of gains between noticeable and uncomfortable for negative rotation gain. In addition, the noticeable gain would most likely have an effect on immersion of the user in the application. Additional research is also required on the topic as it is not certain how using noticeable gains over a longer period of time would affect simulator sickness compared to non-noticeable gains.

7 Conclusion

In this thesis, another measurement of the detection thresholds for rotation, translation and curvature gain was performed. The detection threshold experiment was performed using pseudo-Two-alternative forced choice with a method of constant stimuli. The results found show that detection threshold for rotation gain was at 13% increased gain and at 21% decreased gain. For translation gain, the detection thresholds were at 19% increased translation and at 5% for decreased rotation. With curvature gain, the detection threshold found would cause the user to walk in a circle with 22m in radius. When comparing the detection threshold with other studies with older hardware. It was found that the amount of increased rotation gain and the amount of decreased translation gain that could be applied was significantly less in these results. Having evaluated other possible factors the hardware change seemed like the most plausible cause for the observed change. The observed changes in detection threshold are believed to be caused by the increased FOV in the HMD. The reason why an increased FOV may cause these changes is that it allows for more accurate evaluation of motion. This will have an effect on how usable redirected walking will be as a VR locomotion method. As the new detection thresholds measured limit the amount of gain that can be applied by steering algorithms more than the previously measured detection thresholds.

In addition, an experiment with incrementing gains was performed. With the intention of finding if there was a range of gains that were noticeable by the participant but not uncomfortable. The results indicate that such a range does exist. Though how large the range is varies both between the gain applied and the participants. While the range was found it is not suggested to use the results as a limit for steering algorithms like the detection thresholds. As the number of experiments done is not enough to provide an accurate value. If a comfort threshold or nausea threshold is found it should be optional to use this extended gain limit if the user desires more redirection. As the noticeable gain would most likely affect immersion within the virtual environment.

7.1 Future Work

While this study has shown that hardware specifications have an effect on the detection threshold the question of the what happens in the future still remains. While the HTC Vive is currently considered a modern piece of equipment the Vive Pro version was just released. Further other HMD that promise to provide even

better specification in terms of resolution and FOV are also being developed such as the PiMax 8k. As such it remains to see if the detection thresholds will change further with even more changes in hardware or if the changes will stagnate. In this case, the software developed and methodology used in this thesis could be reused to calculate the detection threshold with new hardware. The software used could also be further improved upon to allow the experiments to be performed without a researcher present. If this was the case the experiment could be released as an application and reach a larger audience for data collection. Thus having the experiments to be automated to such as degree would drastically reduce the amount of time the researcher would need to spend on the experiments. In addition, the incrementing experiment and detection threshold experiment could be separated so that they no longer come in order, but allows the user to chose the experiment. The application developed in this thesis could, in theory, be used to look for a comfort threshold or nausea threshold using pseudo-2AFC. Though this might not be possible as discussed due to simulator sickness build up. As such a different approach for measuring those thresholds might be required. In addition, there is still the question if noticeable gains can be used over a longer period of time or if they should just be used in short intervals. The application developed in this thesis could possibly be used to measure this by increasing the time between each increment, but a new experiment is probably required.

Bibliography

[1] Interrante, V., Ries, B., & Anderson, L. 2007. Seven league boots: A new metaphor for augmented locomotion through moderately large scale immersive virtual environments. In *3D User Interfaces, 2007. 3DUI'07. IEEE Symposium on*. IEEE.

[2] Usoh, M., Arthur, K., Whitton, M. C., Bastos, R., Steed, A., Slater, M., & Brooks Jr, F. P. 1999. Walking> walking-in-place> flying, in virtual environments. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 359–364. ACM Press/Addison-Wesley Publishing Co.

[3] 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 *3D User Interfaces (3DUI), 2015 IEEE Symposium on*, 3–10. IEEE.

[4] Ruddle, R. A. & Lessels, S. 2009. The benefits of using a walking interface to navigate virtual environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(1), 5.

[5] Peck, T. C., Fuchs, H., & Whitton, M. C. 2012. The design and evaluation of a large-scale real-walking locomotion interface. *IEEE transactions on visualization and computer graphics*, 18(7), 1053–1067.

[6] Razzaque, S. 2005. *Redirected walking*. University of North Carolina at Chapel Hill.

[7] Steinicke, F., Bruder, G., Jerald, J., Frenz, H., & Lappe, M. 2010. Estimation of detection thresholds for redirected walking techniques. *IEEE transactions on visualization and computer graphics*, 16(1), 17–27.

[8] Williams, B., Narasimham, G., Rump, B., McNamara, T. P., Carr, T. H., Rieser, J., & Bodenheimer, B. 2007. Exploring large virtual environments with an hmd when physical space is limited. In *Proceedings of the 4th symposium on Applied perception in graphics and visualization*, 41–48. ACM.

[9] Engel, D., Curio, C., Tcheang, L., Mohler, B., & Bülthoff, H. H. 2008. A psychophysically calibrated controller for navigating through large environments in a limited free-walking space. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, 157–164. ACM.

[10] Grechkin, T., Thomas, J., Azmandian, M., Bolas, M., & Suma, E. 2016. Revisiting detection thresholds for redirected walking: Combining translation and curvature gains. In *Proceedings of the ACM Symposium on Applied Perception*, 113–120. ACM.

[11] Hollerbach, J. M. 2002. Locomotion interfaces. *Handbook of virtual environments: Design, implementation, and applications*, 239–254.

[12] Frommel, J., Sonntag, S., & Weber, M. 2017. Effects of controller-based locomotion on player experience in a virtual reality exploration game. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 30. ACM.

[13] Pausch, R., Burnette, T., Brockway, D., & Weiblen, M. E. 1995. Navigation and locomotion in virtual worlds via flight into hand-held miniatures. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 399–400. ACM.

[14] Stoakley, R., Conway, M. J., & Pausch, R. 1995. Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 265–272. ACM Press/Addison-Wesley Publishing Co.

[15] Bertin, R., Israël, I., & Lappe, M. 2000. Perception of two-dimensional, simulated ego-motion trajectories from optic flow. *Vision Research*, 40(21), 2951–2971.

[16] Langbehn, E., Lubos, P., Bruder, G., & Steinicke, F. 2017. Bending the curve: Sensitivity to bending of curved paths and application in room-scale vr. *IEEE transactions on visualization and computer graphics*, 23(4), 1389–1398.

[17] Azmandian, M., Grechkin, T., Bolas, M., & Suma, E. 2016. The redirected walking toolkit: a unified development platform for exploring large virtual environments. In *Everyday Virtual Reality (WEVR), 2016 IEEE 2nd Workshop on*, 9–14. IEEE.

[18] Zmuda, M. A., Wonser, J. L., Bachmann, E. R., & Hodgson, E. 2013. Optimizing constrained-environment redirected walking instructions using search

techniques. *IEEE transactions on visualization and computer graphics*, 19(11), 1872–1884.

[19] Nescher, T., Huang, Y.-Y., & Kunz, A. 2014. Planning redirection techniques for optimal free walking experience using model predictive control. In *3D User Interfaces (3DUI), 2014 IEEE Symposium on*, 111–118. IEEE.

[20] Neth, C. T., Souman, J. L., Engel, D., Kloos, U., Bulthoff, H. H., & Mohler, B. J. 2012. Velocity-dependent dynamic curvature gain for redirected walking. *IEEE transactions on visualization and computer graphics*, 18(7), 1041–1052.

[21] Bruder, G., Steinicke, F., Hinrichs, K. H., Frenz, H., & Lappe, M. 2009. Impact of gender on discrimination between real and virtual stimuli. In *Workshop on Perceptual Illusions in Virtual Environments*, 10–15.

[22] Waldow, K., Fuhrmann, A., & Grünvogel, S. M. 2018. Do textures and global illumination influence the perception of redirected walking based on translational gain? In *IEEE VR Posters*. IEEE.

[23] Meyer, F., Nogalski, M., & Fohl, W. 2016. Detection thresholds in audio-visual redirected walking. In *Proc. Sound and Music Comp. Conf.(SMC)*, volume 16, 17–27.

[24] Suma, E. A., Clark, S., Krum, D., Finkelstein, S., Bolas, M., & Warte, Z. 2011. Leveraging change blindness for redirection in virtual environments. In *Virtual Reality Conference (VR), 2011 IEEE*, 159–166. IEEE.

[25] Steinicke, F. & Bruder, G. 2013. Using perceptual illusions for redirected walking. *IEEE computer graphics and applications*, 33(1), 6–11.

[26] Bruder, G., Steinicke, F., & Wieland, P. 2011. Self-motion illusions in immersive virtual reality environments. In *Virtual Reality Conference (VR), 2011 IEEE*, 39–46. IEEE.

[27] Zank, M. & Kunz, A. 2016. Eye tracking for locomotion prediction in redirected walking. In *3D User Interfaces (3DUI), 2016 IEEE Symposium on*, 49–58. IEEE.

[28] Lang, B. 2018. Researchers exploit natural quirk of human vision for hidden redirected walking in vr. https://www.roadtovr.com/researchers-exploit-natural-quirk-of-human-vision-saccade-hidden-redirect Accessed 31. May 2018.

[29] Azmandian, M., Grechkin, T., & Rosenberg, E. S. 2017. An evaluation of strategies for two-user redirected walking in shared physical spaces. In *Virtual Reality (VR), 2017 IEEE*, 91–98. IEEE.

[30] Hodgson, E., Bachmann, E., & Waller, D. 2011. Redirected walking to explore virtual environments: Assessing the potential for spatial interference. *ACM Transactions on Applied Perception (TAP)*, 8(4), 22.

[31] Langbehn, E. & Steinicke, F. 2018. Redirected walking in virtual reality.

[32] Steinicke, F., Bruder, G., Jerald, J., Frenz, H., & Lappe, M. 2008. Analyses of human sensitivity to redirected walking. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, 149–156. ACM.

[33] Paludan, A., Elbaek, J., Mortensen, M., Zobbe, M., Nilsson, N. C., Nordahl, R., Reng, L., & Serafin, S. 2016. Disguising rotational gain for redirected walking in virtual reality: Effect of visual density. In *Virtual Reality (VR), 2016 IEEE*, 259–260. IEEE.

[34] Neira, C. C., Kiyokawa, K., & Roberts, D. 2012. Redirected steering for virtual self-motion control with a motorized electric wheelchair. *Joint Virtual Reality Conference of ICAT - EGVE - EuroVR*.

[35] Bruder, G., Interrante, V., Phillips, L., & Steinicke, F. 2012. Redirecting walking and driving for natural navigation in immersive virtual environments. *IEEE transactions on visualization and computer graphics*, 18(4), 538–545.

[36] Steinicke, F., Bruder, G., Ropinski, T., & Hinrichs, K. 2008. Moving towards generally applicable redirected walking. In *Proceedings of the Virtual Reality International Conference (VRIC)*, 15–24. IEEE Press.

[37] Chen, K. B., Ponto, K., Sesto, M. E., & Radwin, R. G. 2014. Influence of altered visual feedback on neck movement for a virtual reality rehabilitative system. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 58, 693–697. SAGE Publications Sage CA: Los Angeles, CA.

[38] 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(3), 203–220.

[39] Games, E. 2018. Unreal engine homepage. https://www.unrealengine.com/en-US/what-is-unreal-engine-4. Accessed 27. May 2018.

[40] Technologies, U. 2018. Unity game engine home page. https://unity3d.com/. Accessed 27. May 2018.

[41] Games, E. 2018. Unreal engine 4: Virtual reality best practices. https://docs.unrealengine.com/latest/INT/Platforms/VR/ContentSetup/. Accessed 27. May 2018.

[42] Rama. 2017. Victory plugin github page. https://github.com/EverNewJoy/VictoryPlugin. Accessed 24. October 2017.

[43] Banton, T., Stefanucci, J., Durgin, F., Fass, A., & Proffitt, D. 2005. The perception of walking speed in a virtual environment. *Presence: Teleoperators & Virtual Environments*, 14(4), 394–406.

[44] Nilsson, N. C., Serafin, S., & Nordahl, R. 2014. Establishing the range of perceptually natural visual walking speeds for virtual walking-in-place locomotion. *IEEE transactions on visualization and computer graphics*, 20(4), 569–578.

A Replication

This appendix chapter contains an overview of features that are needed for replication but were not considered important enough to mention in the implementation section. The Github repository for the application is available through the following link: https://github.com/chromekard/master-fov-redirecedwalking

Tools

For the implementation of this project a verity of tools were used. This section contains an overview of the different tool and describes why they were used, how they were used and what possible alternatives there were.

Git

Git was used as the version control system when developing using the unreal engine. Again the reason that git was used was because the Author had more experience with it compared to other version control systems such as Mercurial. When working with the unreal engine it is generally best practice to use source control system in the unreal engine editor. This is the case as blueprints are binary files, as such cannot be merged properly if changes were committed on a file from two different sources. To avoid this the unreal engine source control system makes it so that only one person can check-out and check-in a file at a given time. However, as there was only a single developer for the repository the source control system in unreal was not used.

When committing, pulling and pushing new changes the GIT bash and GUI were generally used. However, the GitHub desktop application [] was also used as when developing on a certain PC the authentication notification would not appear when using Git Bash and GUI.

GitHub

As the hosting service for the unreal engine project GitHub was used. Bitbucket was also a possible alternative to use however the main differences between the two is that Bitbucket allows for an unlimited number of private repositories for small teams while git hub does not. The reason for using GitHub over Bitbucket was that a private repository was not needed for this project. Further from experience it is more common to host open source project on GitHub.

Google Forms

As mentioned in Method 3 the questionnaire used was given on paper. The questionnaire was originally meant to be answered through Google Forms as this was a quick way to create and host a questionnaire. however because of issues regarding NSD and their rules regarding using 3D parties and the storage of personal information the questionnaire was done on paper instead. The Questionnaire was however still made in Google Forms before being printed. A possible options to get around the issue was to host the questionnaire in Questback. As it was provided by the school and follows NSD regulations. However this was evaluated to be to time consuming compared to printing the already created questionnaire from the Google form template.

Experiment Application

This section will cover the implementation that went into developing the experiment application but was not important enough to mention in the Implementation Chapter.

Assets

The assets used when creating the application were all free assets provided by Epic Games the developers of the unreal engine. From the game engine the following packages were used:

- Starter Content
    - From the Starter Content the assets that were used mainly consisted of static meshes such as chairs, tables, lamps and basic shapes such as circles and cubes. As well as some of the particle effects and audio.
- Virtual reality blueprints features
    - From the Virtual reality blueprint features the only asset that was needed was the static mesh of the HTC Vive controllers.
- First Person Shooter blueprints features
    - Same as with the Virtual reality blueprint features only a Gun skeletal mesh was needed from the First Person Shooter blueprints features.

The main assets used in the building the environment came from the blueprint tutorial. To use the assets from the blueprint tutorial unreal engine project the assets were migrated thought the unreal engine editor to the project used for creating the experiment application. The assets were them moved into their own folder from the root folder to keep assets contained. This was done in the editor to ensure that references between assets were not lost. As there assets were the primary assets

used and the migration process slightly bothersome the assets from the tutorial project were included in the project repository. The other assets provided by the game engine however were not included in the repository to avoid bloating. Last a few assets were created by the author such as the cannons and basketball hoop used in the incriminating gain part of the experiment. These static mesh assets were created using the geometry in the editor to combine basic shapes to a static mesh. In addition the author also made some textures and recorded the voice instructions.

Audio

The audio used in the experiment application consisted of the audio instructions and some background noise. The purpose of the background noise was mainly to drown out any outside interference as such it would always be on. The sound used for the background noise was the Starter_Background_Cue from starter content. This sound cue featured wind blowing with birds chirping. While maybe not the most in theme background noise to the environment it was pleasant and not to disturbing. As for the reason behind audio instructions, when performing a miniature version of the experiment before this audio instructions were a much requested feature in addition to the text instructions. As such the audio instructions were all recorded by the author before being imported into the unreal project. The audio recordings were specifically 16 bit Wave files sampled at 44100 Hz.

Targets

The targets used consists of a single plane mesh component. The plane is set to generate hit events when colliding with another object. In the case that object is a bullet a time line is then used to set a scalar parameter in the plane meshes dynamic material. The material is made so that when the scalar parameter is changed the material will disintegrate depending on the value given. The disintegration material is base on the following tutorial: https://www.youtube.com/watch?v=gldIJGqlWf0. A delay is then used to reset the time line after the time line was initially completed.

Balls

Ball objects were created to be grab able by player. To do this the ball class implements the grab interface. The grab interface contains two functions. One for grabbing the ball that provides the component to attach the ball to, and a second function for releasing the actor from the attached component. The ball objects themselves only consist of a single sphere static mesh. Which then uses a material that has a vector parameter values for controlling the color. This is used in the constructor to randomize the color of the ball.

Cannon

The cannon object consists of a cannon static mesh, a scene component at the muzzle of the cannon, and a text render component for displaying the score. When the cannon is created is random count down is set. In the tick function time is aggregated until the countdown is reached. When reached a new count down will be randomly generated and a bomb object is spawned at the muzzle scene component. An event is bound to the spawned bombs explode event. In the case that the explode event returns true the score is reduced by 1 in case if not the score is increased by 1. To move the cannons a time line is used to linearly interpolate the yaw of the cannon.

The bomb objects uses the same material as the ball objects. However when created there is the 15% chance that it is a bomb and a 85% chance that it is a dud. In the case the object is a dud its color is green if not then black. I the user collides with the sphere static mesh or the sphere collision component of the object the explode event is called and the object disappears. To get the correct amount of movement for the player to be able to catch the object the projectile movement component was adjusted, so that only a tenth the usual gravity is applied to the object as well as setting a low initial speed.

B Instructions

This appendix chapter contain the written instructions used in the tutorial of the experiment, the measurement of eye height and gait, increment experiment and detection threshold experiment. Each are enumerated in the order in which they were presented.

Tutorial

1. Welcome to the introduction of this experiment. To see what the different buttons on the controllers do press the side buttons on the left-hand controller. Note that what the buttons on the right controller might change their function depending on the stage of the experiment. As you may have noticed on the left hand are the written instructions as well as the user interface for controlling the audio instructions. In addition, there are also buttons for starting the experiments and ending them. Try pressing the begin button.
2. Good, when a stage of the experiment is finished the finish buttons will appear. When you feel ready to proceed, press the Finish button to start the next stage of the experiment.

Measurement

1. In this part of the experiment, the intention is to measure your gait and height. To measure your gait hold down the top part of the right controller's trackpad and take two steps. To measure your height press down the bottom part of the right controller's trackpad while standing straight and looking forward. Be sure to press the begin button to start the measurements.
2. Great the measurements have been recorded. If you wish to feel free to remeasure. Press the finish button when you are ready to progress to the next stage of the experiment.

Increment experiment

1. In this stage of the experiment, the intention is to measure the point at which you notice the gain being applied. For the gain to be applied it is required that you move around within the room as it is impossible to notice while standing still. When you notice the gain press the top part of the trackpad on the right controller. In case you don't notice the gain the experiment will

progress by itself after a given time. Remember to press the being button to start the experiment.
2. You were able to notice that the gain was applied. The next part of the experiment is to find your limit. As such simply press the same button again if the experience becomes too jarring. And remember the experiment will progress by itself automatically at some point. Press the continue button to proceed.
3. You have now finished the first half of this stage of the experiment. The next part will consist of the same stages as before but with opposite gains. As such do the same as in the beginning and press the up button on the right trackpad when you notice the gain being applied. Press the continue button to progress.
4. You were able to notice that the gain was applied. The next part of the experiment is to find your limit. As such simply press the same button again if the experience becomes too jarring. And remember the experiment will progress by itself automatically at some point. Press the continue button to proceed.
5. You have now completed this part of the experiment press the finish button to go on to the next experiment.

Detection threshold experiment

Rotation Gain

1. In this part of the experiment, the purpose is to measure how noticeable different intensities of gain are. When you start the experiment target for you to look at will be placed at 90 degrees to either side of you. For each gain intensity, you are required to look at one of these targets before evaluating if the rotation in the virtual world was larger or less than the rotation in the physical world. To simplify if you notice that you have to rotate a large amount or that the rotation is slower than normal then press down on the right trackpad and if you don't have to rotate much and the rotation is faster than normal then press up on the right trackpad. If you are uncertain then just randomly press up or down. Then repeat for the remaining gain instances. The number of gain instances remaining can be seen above the left controller.
2. Fantastic, you have completed this experiment. Press the Finish button to save your results and exit the program.

Translation Gain

1. Before starting this part of the experiment the virtual and physical space needs to be aligned. As such give the VR headset temporarily to the instructor.
2. In this part of the experiment, the purpose is to measure how noticeable different increases and decreases of movement are. As such first use the side

buttons on the right controller to indicate that you are in position. A green ball will then appear ahead of you. Then walk towards it until it turns blue. Then use the up button on the trackpad on the right controller if you noticed that you moved shorter in the physical world than the virtual world or press the down button if you noticed that you moved longer in the physical world compared to the virtual world. To simplify if you notice that you move faster than normal press up or if you move slower than normal press down. Then repeat going back to the start position, pressing the in position button, walking towards the green ball and then pressing up or down. The number of times remaining can be seen above the left controller.

3. Fantastic, you have completed this experiment. Press the Finish button to save your results and exit the program.

Curvature Gain

1. Before starting this part of the experiment the virtual and physical space needs to be aligned. As such give the VR headset temporarily to the instructor.
2. In this part of the experiment, the purpose is to measure how noticeable different curvature intensities are. As such first use the side buttons on the right controller to indicate that you are in position. A green ball will then appear ahead of you. Then walk towards it until it turns blue. Then use the up button on the trackpad on the right controller if you noticed any curving or press the down button if you did not notice any curving. Then repeat going back to teh start position, pressing the in possition button, walking towards the green ball and then pressing up or down. The number of times remaining can be seen above the left controller.
3. Fantastic, you have completed this experiment. Press the Finish button to save your results and exit the program.

C Questionnaire

The following pages contain the pdf of the questionnaire given to participants in the experiment.

Demographic Questions

Some questions about you the paritcipant

* Required

1. **Id** *

_____

2. **Gender** *

Mark only one oval.

- ◯ Female
- ◯ Male
- ◯ Prefer not to say
- ◯ Other: _____

3. **Age** *

Mark only one oval.

- ◯ < 20
- ◯ 20 - 30
- ◯ 30 - 40
- ◯ 40 - 50
- ◯ > 50

4. **Do you use anything for vision correction?** *

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe
- ◯ Prefer not to say

5. **If you are using corrective lenses are you able to wear them in the VR headset?**

Mark only one oval.

- ◯ Yes
- ◯ No

6. **How much digital game experience do you have?** *

Mark only one oval.

- ◯ I have not played any digital games
- ◯ I have played a small variety of digital games on a few platforms
- ◯ I have played a variety of digital games on several platforms
- ◯ I have played a large variety of digital games on most platforms

7. **How much VR experience do you have?** *
Mark only one oval.

- I have not tried a VR headset
- I have tried a VR headset once for a few minutes
- I have used a VR headset a couple of time for a few minutes
- I have used a VR headset multiple times and used it for a long session

8. **Do you believe there is a reason that your results would not be valid in this study?** *
Mark only one oval.

- Yes
- No
- Maybe

9. **In case of yes or maybe was given above would you please elaborate?**

Simulator sickness pre test

A set of questions to get a benchmark on your current degree of simulator sickness.

10. **Questions** *
Mark only one oval per row.

	None	Slight	Moderate	Severe
General Discomfort				
Fatigue				
Headache				
Eye strain				
Difficulty focusing				
Salivation increasing				
Sweating				
Nausea				
Difficulty concentrating				
« Fullness of the Head »				
Blurred vision				
Dizziness with eyes open				
Dizziness with eyes closed				
Vertigo				
Stomach awareness				
Burping				

Simulator sickness post test

Do these questions after you have completed the VR test.

11. **Questions** *

Mark only one oval per row.

	None	Slight	Moderate	Severe
General Discomfort				
Fatigue				
Headache				
Eye strain				
Difficulty focusing				
Salivation increasing				
Sweating				
Nausea				
Difficulty concentrating				
« Fullness of the Head »				
Blurred vision				
Dizziness with eyes open				
Dizziness with eyes closed				
Vertigo				
Stomach awareness				
Burping				

D Raw Data

This appendix contains additional data from the experiments that was considered to long for the results chapter. This includes the individual detection thresholds for rotation9, translation 10 and curvature gain 11. It also includes the answers given during the detection threshold experiment for rotation 15, translation 16 and curvature 17 gain. The individual results from the incriminating gain experiments are in Table 12 for rotation gain, Table 13 for translation gain and Table 14 for curvature gain. An example of the tracking data gathered can be found in Table 27. As the tracking data is to large to have every file included in the appendix, the remaining files are hosted at https://drive.google.com/drive/folders/1AQgLjs4QxMYvAcTfnGW8MGckMs1F6869?usp=sharing

Participants

The individual results from the questionnaire for rotation 18, translation 19 and curvature 20 gain are included bellow. The answers to the multiple choice questions are written as numbers 1-4 instead of the answer from the questionnaire to save space. Meaning 1 is the first option and 4 is the last. The last "Comp" column was added to specify if the user completed the experiment completely or not. The results from the Simulation sickness questionnaire (SSQ) have also been included. The results were shortened to their Nausea, Oculomotor, Disorientation weights and the total SSQ score. The tables of the SSQ results for rotation 21, translation 22 and curvature 23 gain are bellow.

Event	x	y	z	roll	pitch	yaw
d	-23.468	4.943	146.184	-1.297	-35.638	-145.403
d	-23.174	6.008	146.460	-5.186	-36.410	-144.449
d	-22.688	5.850	146.741	-4.310	-36.475	-143.655
d	-22.438	6.724	146.723	-2.604	-36.399	-147.017
d	-21.137	6.747	147.583	-4.352	-33.564	-145.495
d	-21.182	6.129	147.604	-2.684	-33.472	-147.155
d	-21.441	5.385	147.212	-2.413	-34.442	-146.853
d	-21.601	5.654	147.206	-2.442	-34.625	-146.554
d	-21.189	5.727	147.141	-3.511	-34.152	-144.309
d	-20.764	6.011	147.105	-2.757	-34.360	-145.162
d	-20.812	6.485	147.037	-2.782	-34.633	-145.735
d	-20.566	6.464	147.107	-2.976	-34.636	-145.074

Event	x	y	z	roll	pitch	yaw
d	-20.356	6.571	147.000	-2.613	-35.207	-145.878
d	-20.088	6.278	146.672	-3.739	-36.087	-143.570
d	-19.929	5.807	146.369	-3.713	-36.969	-143.218
d	-19.197	5.086	147.085	-1.014	-35.515	-145.954
d	-18.449	5.305	148.355	1.116	-34.107	-153.967
d	-19.275	6.615	147.907	4.105	-37.514	-160.899
d	-20.804	9.337	148.322	4.529	-37.203	-168.512
d	-20.950	9.622	148.392	3.172	-37.591	-165.911
d	-20.901	9.640	148.761	4.486	-36.280	-165.535
d	-20.100	9.577	148.927	3.249	-35.816	-162.972
d	-19.721	9.754	148.905	-1.225	-35.856	-160.078
d	-19.989	10.045	149.249	-2.814	-34.812	-159.938
d	-20.274	10.243	149.715	-3.959	-32.903	-161.269
d	-20.324	9.781	149.987	-3.830	-31.795	-160.125
d	-20.382	9.005	149.999	-3.448	-31.733	-157.605
d	-20.599	8.829	150.102	-3.597	-31.298	-157.285
d	-20.646	8.924	150.284	-4.714	-30.578	-156.970
d	-20.814	9.304	150.316	-5.367	-30.517	-156.074
d	-20.701	9.426	150.529	-6.049	-29.620	-155.639
d	-21.062	9.473	150.465	-7.772	-29.992	-156.588
d	-21.550	9.951	150.240	-10.571	-30.913	-160.916
d	-21.763	10.435	149.982	-9.902	-31.743	-165.463
d	-21.665	10.196	149.757	-9.649	-32.584	-164.676
d	-21.569	9.716	149.588	-9.304	-33.189	-162.823
d	-21.595	9.613	148.959	-9.022	-36.067	-161.879
d	-21.767	9.488	148.896	-8.677	-36.324	-162.269
d	-22.033	9.750	149.054	-9.219	-35.807	-162.130
d	-22.785	9.893	148.779	-9.170	-36.269	-164.694
d	-22.659	7.701	149.134	-6.921	-33.423	-159.698
d	-22.224	3.769	150.218	0.548	-26.015	-143.980
d	-20.141	0.617	150.725	8.372	-21.013	-132.481
d	-19.759	0.819	151.290	10.082	-18.664	-134.213
d	-20.416	2.018	151.722	9.514	-16.053	-137.362
d	-20.827	3.076	151.826	9.043	-15.147	-139.671
d	-21.054	3.763	151.551	7.162	-15.885	-140.145
d	-22.374	3.183	148.832	1.863	-21.854	-137.524
d	-23.146	3.083	146.003	-4.915	-29.596	-133.746
d	-22.658	3.798	147.006	-5.104	-26.000	-135.903
d	-21.498	4.630	148.841	-6.577	-20.217	-134.950
d	-21.329	5.358	148.988	-5.634	-19.705	-138.325
d	-21.412	5.462	149.151	-4.023	-20.095	-141.421
d	-21.092	6.342	149.476	-3.271	-21.318	-145.661
d	-20.722	6.897	149.578	-4.040	-22.855	-148.161
d	-20.890	7.161	149.610	-3.548	-22.845	-149.259

Event	x	y	z	roll	pitch	yaw
d	-21.274	7.373	149.819	-3.110	-21.441	-150.837
d	-20.791	7.556	150.231	-2.472	-20.780	-151.808
d	-20.261	7.903	150.491	-0.039	-21.381	-155.443
d	-19.914	7.426	150.789	1.957	-21.706	-156.293
d	-20.107	7.452	150.868	2.381	-22.123	-156.372
d	-19.803	8.621	149.125	3.867	-32.827	-160.581
d	-20.013	8.487	148.132	3.965	-38.520	-160.853
d	-20.966	9.439	148.463	4.261	-37.552	-165.754
d	-21.404	10.335	149.283	-3.548	-34.783	-165.615
d	-21.783	10.114	149.320	-6.721	-34.485	-163.354
d	-22.309	9.778	149.214	-7.858	-34.714	-162.758
d	-22.454	9.729	149.210	-8.854	-34.443	-162.917
d	-22.533	9.917	149.414	-9.784	-33.665	-164.704
d	-22.221	9.449	149.580	-10.028	-32.906	-162.684
d	-22.346	9.602	149.351	-9.056	-34.208	-164.883
d	-22.384	9.885	149.061	-8.716	-35.476	-165.970
d	-22.529	10.746	149.121	-11.444	-34.935	-164.211
d	-22.580	12.081	149.088	-12.258	-34.721	-164.521
d	-22.564	10.778	149.246	-11.302	-34.213	-159.886
d	-23.574	10.229	149.017	-10.890	-35.424	-161.459
d	-22.552	10.816	148.979	-7.880	-35.590	-167.400
d	-23.701	7.353	145.408	-6.535	-48.144	-163.926
d	-24.123	1.826	141.373	-21.307	-60.443	-131.263
d	-23.191	1.823	142.671	-25.762	-55.784	-120.529
d	-23.468	3.874	144.900	-22.952	-47.966	-124.639
d	-23.023	5.465	145.845	-20.244	-44.551	-128.900
d	-22.914	6.287	145.320	-14.631	-47.978	-143.601
d	-22.682	9.208	145.866	-1.804	-47.513	-167.398
d	-22.262	12.596	146.511	0.457	-44.716	176.774
d	-22.387	11.905	146.728	-2.577	-44.794	-178.552
d	-22.503	10.632	146.265	-3.977	-46.805	-169.436
d	-22.660	10.720	146.142	-2.671	-46.862	-168.429
d	-23.024	11.502	145.869	-0.156	-48.108	-173.403
d	-23.324	11.680	145.326	-1.403	-50.567	-174.455
d	-23.318	11.446	145.206	-3.013	-51.088	-172.292
d	-23.430	10.918	145.622	-4.693	-49.843	-169.132
d	-22.847	10.306	145.685	-4.748	-49.755	-166.631
d	-22.108	9.253	144.983	-5.220	-51.726	-159.858
d	-21.252	6.100	143.097	-11.365	-56.631	-139.244
d	-20.559	4.374	141.110	-16.248	-62.450	-128.931
d	-20.766	4.102	141.144	-10.007	-63.537	-133.712
d	-19.878	3.379	142.733	-9.510	-60.980	-130.849
d	-19.017	3.506	143.603	-4.622	-56.876	-135.293
d	-19.716	5.011	145.074	-3.657	-51.351	-139.188

Event	x	y	z	roll	pitch	yaw
d	-20.736	7.169	145.818	-2.768	-49.545	-153.280
d	-21.381	8.549	146.012	-8.639	-48.923	-157.399
d	-21.927	9.073	146.670	-7.948	-46.024	-156.304
d	-22.244	8.634	146.835	-6.142	-45.207	-154.922
d	-22.121	8.059	147.160	-7.137	-44.025	-152.448
d	-22.224	7.864	146.854	-6.940	-45.233	-153.921
d	-22.488	8.006	147.112	-6.994	-44.313	-155.304
d	-22.478	8.150	147.578	-7.019	-42.549	-155.738
d	-22.335	8.217	148.731	-5.136	-37.811	-157.023
d	-22.009	8.189	148.569	-5.056	-38.556	-156.392
d	-21.839	8.361	148.233	-6.030	-40.027	-155.806
d	-21.625	8.553	147.902	-6.807	-41.176	-155.279
d	-21.469	8.866	147.769	-7.003	-41.732	-155.964
d	-21.378	9.131	147.614	-7.037	-42.315	-156.239
d	-21.378	9.191	147.483	-7.151	-42.886	-156.331
d	-21.553	9.194	147.452	-7.019	-43.013	-156.228
d	-21.615	9.048	147.360	-7.100	-43.539	-155.269
d	-21.595	8.836	147.302	-7.323	-43.729	-154.310
d	-21.744	8.922	147.342	-7.504	-43.696	-154.222
d	-21.817	8.995	147.286	-7.524	-43.826	-155.285
d	-22.026	9.086	147.213	-7.481	-44.041	-155.839
d	-21.448	7.826	146.952	-8.051	-44.799	-148.308
d	-21.166	7.043	146.603	-7.333	-46.403	-144.346
d	-20.908	6.364	146.283	-6.769	-47.942	-142.472
d	-20.264	5.502	146.365	-8.037	-47.390	-136.773
d	-20.338	5.589	146.333	-6.826	-47.581	-138.298
d	-20.112	5.657	146.341	-7.400	-47.376	-137.265
d	-19.748	5.505	146.363	-7.149	-47.341	-137.054
d	-19.748	5.711	146.561	-6.870	-46.694	-138.103
d	-19.805	5.925	146.621	-7.140	-46.462	-138.456
d	-19.909	5.878	146.615	-7.385	-46.355	-138.930
d	-20.156	5.802	146.705	-7.264	-46.165	-139.572
d	-20.611	6.336	146.950	-7.514	-45.381	-142.777
d	-21.001	7.077	147.056	-5.602	-45.173	-146.888
d	-21.122	7.345	146.972	-5.935	-45.398	-148.401
d	-21.033	7.231	146.852	-6.163	-45.823	-147.749
d	-21.184	7.543	146.925	-6.228	-45.720	-148.361
d	-21.260	8.015	147.038	-7.104	-45.264	-148.919
d	-21.308	8.390	147.578	-5.816	-43.225	-152.522
d	-20.980	8.165	147.891	-5.105	-41.914	-152.134
d	-20.775	8.075	148.224	-5.247	-40.563	-151.919
d	-20.906	8.552	147.734	-3.742	-42.605	-155.930
d	-21.268	8.659	147.409	-4.266	-43.812	-156.426
d	-21.672	8.690	147.407	-4.816	-43.870	-156.868

Event	x	y	z	roll	pitch	yaw
d	-22.104	8.567	147.700	-4.842	-42.584	-157.728
d	-22.443	8.313	147.890	-4.941	-42.092	-157.432
d	-22.486	8.122	147.805	-5.061	-42.345	-156.790
d	-22.399	8.071	147.671	-5.310	-42.781	-156.155
d	-22.376	8.265	147.616	-5.544	-42.955	-155.952
d	-22.213	8.577	147.613	-5.716	-42.849	-155.956
d	-21.995	8.855	147.625	-5.926	-42.833	-156.268
d	-21.718	9.108	147.733	-6.119	-42.270	-156.110
d	-21.435	9.391	147.876	-6.361	-41.525	-156.109

Table 27: Tracking data of participant C03 from the tutorial part of the experiment

Tick	Virtual rotation	Smoother virtual rotation
1	0.017132	-0.237151
2	0.009891	-0.11544
3	-0.013988	-0.070684
4	-0.027976	-0.052827
5	-0.052339	-0.058674
6	-0.085087	-0.080067
7	-0.105145	-0.09762
8	-0.170174	-0.150155
9	-0.212603	-0.191986
10	-0.290404	-0.260646
11	-0.368507	-0.334102
12	-0.431944	-0.398882
13	-0.570019	-0.518969
14	-0.663608	-0.614686
15	-0.851304	-0.779919
16	-0.861133	-0.822983
17	-0.929127	-0.893053
18	-1.168872	-1.090899
19	-1.306497	-1.233104
20	-1.409799	-1.347277
21	-1.172884	-1.200852
22	-1.460739	-1.402759
23	-1.658382	-1.579982
24	-1.706701	-1.655421
25	-1.869292	-1.803004
26	-1.928508	-1.88056
27	-2.080945	-2.018861
28	-1.863837	-1.887072

29	-1.996216	-1.974739
30	-2.046282	-2.023027
31	-2.014768	-2.011019
32	-2.290047	-2.219353
33	-1.865963	-1.936637
34	-2.138088	-2.105393
35	-1.901307	-1.944155
36	-1.893748	-1.917062
37	-1.883848	-1.89798
38	-2.004952	-1.981742
39	-1.976375	-1.971914
40	-1.94383	-1.949736
41	-2.131689	-2.087677
42	-1.855149	-1.902278
43	-2.103992	-2.065346
44	-1.755951	-1.823638
45	-1.84151	-1.853964
46	-1.879138	-1.875958
47	-1.902157	-1.894812
48	-1.787803	-1.812719
49	-1.872745	-1.863968
50	-1.831915	-1.837734
51	-1.67322	-1.715804
52	-1.689783	-1.706934
53	-1.822056	-1.797563
54	-1.298532	-1.417167
55	-1.575971	-1.565929
56	-1.447513	-1.474606
57	-1.694353	-1.64619
58	-1.155018	-1.26577
59	-1.481927	-1.455576
60	-1.317314	-1.345292
61	-1.272264	-1.297516
62	-1.385323	-1.369684
63	-1.170964	-1.216734
64	-1.281309	-1.276608
65	-1.349826	-1.330346
66	-1.43941	-1.407274
67	-1.111144	-1.177142
68	-1.485226	-1.424705
69	-1.280851	-1.301684
70	-1.307582	-1.311316
71	-1.400395	-1.379059
72	-1.55345	-1.504519
73	-1.135924	-1.215839

74	-1.393459	-1.369033
75	-1.502719	-1.463191
76	-1.613111	-1.565749
77	-1.39623	-1.426769
78	-1.493375	-1.484359
79	-1.607975	-1.574817
80	-1.581984	-1.571903
81	-1.210617	-1.298418
82	-1.45207	-1.435607
83	-1.499524	-1.479429
84	-1.452171	-1.453962
85	-1.328743	-1.360495
86	-1.386664	-1.38806
87	-1.370656	-1.375356
88	-1.213726	-1.255308
89	-1.214814	-1.235333
90	-1.341243	-1.319895
91	-0.909004	-1.00639
92	-1.205676	-1.180201
93	-0.81819	-0.902324
94	-0.846868	-0.881765
95	-0.794034	-0.824691
96	-0.741199	-0.769736
97	-0.697467	-0.722669
98	-0.402393	-0.488762
99	-0.455643	-0.485515
100	-0.332792	-0.378441
101	-0.335428	-0.357593
102	-0.192052	-0.238979
103	-0.147375	-0.182008
104	-0.098912	-0.128344
105	-0.017132	-0.052293
106	0.034264	0.003835
107	0.13417	0.093979
108	0.176108	0.145528
109	0.214208	0.189393
110	0.314659	0.277139
111	0.402393	0.361699
112	0.4408	0.410851
113	0.502513	0.47211
114	0.548054	0.521468
115	0.600284	0.573933
116	0.66006	0.631941
117	0.574805	0.582059
118	0.650805	0.635432

119	0.638588	0.633955
120	0.583841	0.595211
121	0.586849	0.591782
122	0.538509	0.553061
123	0.488994	0.508649
124	0.489794	0.499421
125	0.300015	0.352274
126	0.31528	0.337593
127	0.251011	0.278234
128	0.204391	0.229658
129	0.134898	0.164904
130	0.088469	0.11508
131	0.043115	0.067759
132	-0.024228	0.004929
133	-0.044235	-0.024654
134	-0.055953	-0.043233
135	-0.058517	-0.051516
136	-0.077253	-0.069068
137	-0.085087	-0.079036
138	-0.093836	-0.088623
139	-0.108352	-0.102117
140	-0.111468	-0.107571
141	-0.106531	-0.105817
142	-0.123541	-0.118931
143	-0.082755	-0.090647
144	-0.082162	-0.086256
145	-0.057675	-0.065844
146	-0.041965	-0.049977
147	-0.045327	-0.048492
Average	-0.8286726054	-0.8318982517
Sum	-121.814873	-122.289043

Table 28: Virtual rotation applied compared to smoothed virtual rotation applied, during 360 virtual turn, with 1.5 rotation gain. Though as mentioned in the implementation this the rotation that is added to the always present 1.0 rotation gain. As such this is actually 0.5 of the real rotation

UserId	Upper Threshold	PSE	Lower Threshold
A02	0.61	1.43	2.25
A03	0.55	0.78	1.02
A04	0.68	0.82	0.96
A05	0.97	1.06	1.15
A06	0.74	0.88	1.02
A07	0.81	0.9	1
A08	0.76	0.78	0.81
A09	0.83	0.99	1.16
A10	0.78	0.97	1.16
A13	0.92	1.01	1.1
A16	0.97	1.06	1.16
A17	0.98	1.12	1.26

Table 9: Individually calculated thresholds for rotation gain

UserId	Upper Threshold	PSE	Lower Threshold
B01	0.91	1.04	1.18
B02	0.96	1.00	1.04
B03	0.81	0.91	1.00
B04	1.12	1.18	1.24
B05	0.96	1.01	1.07
B06	1.00	1.09	1.18
B07	1.09	1.13	1.17
B08	0.55	0.80	1.04
B09	1.02	1.13	1.23
B10	1.14	1.21	1.28
B11	0.96	1.09	1.22

Table 10: Individually calculated thresholds for translation gain

UserId	Detection threhold
C01	0.0925
C02	0.0232
C03	0.0633
C04	0.0188
C05	0.018
C06	-
C07	0.028
C08	0.0281
C09	0.0349
C10	0.0757
C11	0.0219
C12	0.0817
C13	0.0392
C14	0.0558

Table 11: Individually calculated threshold for curvature gain, negative changed to positive

UserID	DetectPos	MxGainPos	DetectNeg	MxGainNeg
A01	1.157	1.998	0.631	0.580
A02	1.529	2.408	0.841	0.721
A03	1.298	1.674	0.885	0.754
A04	2.525	4.850	0.154	0.154
A05	1.500	1.815	0.641	0.365
A06	1.441	1.850	0.663	0.663
A07	2.326	3.078	0.499	0.054
A08	1.263	1.821	0.842	0.791
A09	-	-	0.808	0.617
A10	1.769	4.892	0.138	0.138
A11	-	-	-	-
A12	2.774	2.831	0.629	0.547
A13	4.050	4.289	0.913	0.332
A14	1.820	1.982	0.864	0.730
A15	1.440	2.249	0.659	0.659
A16	1.146	1.301	0.948	0.859
A17	1.492	1.792	0.925	0.855
Average	1.793	2.552	0.690	0.551

Table 12: At which gains the different participants noticed the rotation gain being applied and at which gain they reached their limit for positive and negative gains.

UserID	DTPositive	MxGainPos	DTNegative	MxGainNeg
B01	1.503	2.194	0.764	0.117
B02	1.659	3.141	0.420	0.145
B03	1.144	2.180	0.755	0.475
B04	1.705	3.583	0.600	0.457
B05	1.177	1.351	0.597	0.597
B06	1.366	1.491	0.908	0.721
B07	1.900	2.737	0.909	0.674
B08	1.383	1.829	0.726	0.675
B09	1.464	1.812	0.903	0.745
B10	2.003	3.086	0.714	0.384
B11	1.098	1.253	0.862	0.804
Average	1.491	2.242	0.742	0.527

Table 13: At which gains the different participants noticed the translation gain being applied and at which gain they reached their limit for positive and negative gain.

UserID	Detection	Limit
C01	0.3820	0.3820
C02	-0.1205	-0.3666
C03	-0.1671	-0.1671
C05	0.0721	0.0873
C06	0.0524	0.3187
C07	0.0666	0.3725
C08	-0.1041	-0.1359
C09	0.0670	0.0790
C10	0.3742	0.3742
C12	0.2959	0.3780
C13	-0.3648	-0.3648
C14	0.3531	0.3531
Average	0.2016	0.2816

Table 14: At which gains the different participants noticed the curvature gain being applied and at which gain they reached their limit.

User ID	Rotation Gain										
	-0.5	-0.4	-0.3	-0.2	-0.1	0.0	0.1	0.2	0.3	0.4	0.5
A02	7	10	7	9	5	4	6	5	6	5	6
A03	8	7	5	5	6	1	2	0	1	2	0
A04	9	10	5	7	3	2	0	1	1	0	0
A05	9	10	10	8	10	5	6	0	0	0	0
A06	10	8	9	7	4	2	3	1	0	0	0
A07	10	8	9	8	5	3	0	0	0	1	0
A08	10	9	10	3	1	3	1	1	0	0	0
A09	10	10	9	7	5	7	3	2	1	0	0
A10	10	10	10	5	6	4	3	2	3	1	0
A13	10	10	10	10	9	3	5	0	0	0	0
A16	10	10	10	10	9	6	5	1	0	1	0
A17	10	10	10	10	9	9	3	1	5	2	1

Table 15: Amount of times individual answered that virtual rotation was less than real rotation for each rotation gain

User ID	Translation Gain										
	-0.4	-0.3	-0.2	-0.1	0	0.1	0.2	0.3	0.4	0.5	0.6
B01	10	10	10	8	4	4	3	1	1	1	0
B02	10	10	10	9	5	0	1	0	0	0	0
B03	10	10	8	4	4	0	1	0	0	0	0
B04	10	10	10	10	10	8	4	1	0	0	0
B05	10	10	10	10	5	2	1	0	0	0	0
B06	10	10	10	9	8	4	2	2	0	0	0
B07	10	10	10	10	10	7	1	1	0	0	0
B08	5	7	6	4	4	0	2	1	0	0	0
B09	9	9	10	8	9	5	4	1	0	0	0
B10	10	10	10	8	10	10	4	3	0	1	0
B11	10	10	10	7	7	5	3	1	1	0	0

Table 16: Amount of times individual answered that virtual movement was less than real movement for each translation gain

User ID	Curvature Gain					
	0	$\frac{\pi}{180}$	$\frac{2\pi}{180}$	$\frac{3\pi}{180}$	$\frac{5\pi}{180}$	$\frac{6\pi}{180}$
C01	10	10	10	10	8	3
C02	6	6	3	4	4	2
C03	10	10	10	8	3	0
C04	4	8	5	0	1	1
C05	10	5	1	1	0	0
C06	10	10	10	10	10	10
C07	8	9	2	2	0	0
C08	10	9	2	1	0	0
C09	10	9	5	1	0	0
C10	10	9	10	9	5	2
C11	10	6	2	0	0	0
C12	10	8	9	8	8	1
C13	10	8	7	1	0	0
C14	10	9	7	7	2	0

Table 17: Amount of times individual answered no curvature for each curvature gain

UserID	Sex	Age	Vis Cor	Lenses	gExp	vrExp	Reason Inv	Comp
A1	M	20-30	TRUE	FALSE	4	4	FALSE	FALSE
A2	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
A3	M	20-30	TRUE	TRUE	4	4	FALSE	TRUE
A4	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
A5	M	20-30	FALSE	FALSE	4	3	FALSE	TRUE
A6	M	20-30	TRUE	TRUE	4	3	FALSE	TRUE
A7	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
A8	M	20-30	TRUE	FALSE	4	4	FALSE	TRUE
A9	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
A10	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
A11	F	20-30	FALSE	FALSE	2	3	FALSE	FALSE
A12	F	20-30	TRUE	TRUE	3	3	FALSE	FALSE
A13	M	20-30	FALSE	FALSE	3	3	FALSE	TRUE
A14	F	20-30	FALSE	FALSE	3	3	FALSE	FALSE
A15	M	20-30	FALSE	FALSE	4	4	FALSE	FALSE
A16	M	20-30	TRUE	TRUE	4	3	FALSE	TRUE
A17	M	20-30	TRUE	FALSE	4	4	FALSE	TRUE

Table 18: Rotation gain questionnaire results

UserID	Sex	Age	Vis Cor	Lenses	gExp	vrExp	Reason Inv	Comp
B1	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
B2	M	20-30	TRUE	TRUE	4	4	FALSE	TRUE
B3	M	20-30	TRUE	FALSE	4	4	FALSE	TRUE
B4	M	20-30	TRUE	TRUE	2	4	FALSE	TRUE
B5	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
B6	M	20-30	TRUE	FALSE	4	4	FALSE	TRUE
B7	M	20-30	FALSE	FALSE	4	3	FALSE	TRUE
B8	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
B9	M	20-30	TRUE	TRUE	4	3	FALSE	TRUE
B10	M	20-30	FALSE	FALSE	3	2	FALSE	TRUE
B11	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE

Table 19: Translation gain questionnaire results

UserID	Sex	Age	Vis Cor	Lenses	gExp	vrExp	Reason Inv	Comp
C1	M	20-30	TRUE	FALSE	4	4	FALSE	TRUE
C2	M	20-30	FALSE	FALSE	2	4	FALSE	TRUE
C3	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
C4	M	20-30	TRUE	TRUE	4	4	FALSE	TRUE
C5	M	20-30	TRUE	TRUE	4	4	FALSE	TRUE
C6	M	20-30	TRUE	FALSE	4	1	FALSE	TRUE
C7	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
C8	M	20-30	FALSE	FALSE	4	4	FALSE	TRUE
C9	M	20-30	TRUE	TRUE	3	4	FALSE	TRUE
C10	M	20-30	FALSE	FALSE	3	3	FALSE	TRUE
C11	M	20-30	FALSE	FALSE	4	1	FALSE	TRUE
C12	M	<20	FALSE	FALSE	3	2	FALSE	TRUE
C13	M	20-30	TRUE	TRUE	4	2	FALSE	TRUE
C14	M	20-30	FALSE	FALSE	3	2	FALSE	TRUE

Table 20: Curvature gain questionnaire results

User ID	Pre				Post			
	Naus	OM	Dis	SSQ-score	Naus	OM	Dis	SSQ-score
A02	0	2	0	7.48	0	2	1	11.22
A03	2	0	0	7.48	2	0	0	7.48
A04	0	1	0	3.74	2	1	1	14.96
A05	0	1	1	7.48	2	4	3	33.66
A06	0	1	0	3.74	1	1	0	7.48
A07	0	1	0	3.74	4	2	2	29.92
A08	2	5	2	33.66	2	7	1	37.4
A09	2	2	2	22.44	1	0	1	7.48
A10	0	0	0	0	1	0	0	3.74
A11	0	2	1	11.22	2	4	1	26.18
A12	0	0	0	0	11	7	6	89.76
A13	2	1	0	11.22	2	2	1	18.7
A14	0	1	0	3.74	4	2	2	29.92
A15	0	1	0	3.74	0	1	0	3.74
A16	2	2	1	18.7	6	5	2	48.62
A17	2	4	2	29.92	7	8	5	74.8
Average				10.52				27.82

Table 21: SSQ results for rotation participants

User ID	Pre				Post			
	Naus	OM	Dis	SSQ-score	Naus	OM	Dis	SSQ-score
B1	0	0	0	0	0	0	2	7.48
B2	1	0	0	3.74	1	1	0	7.48
B3	0	0	0	0	3	4	3	37.4
B4	1	2	1	14.96	0	2	1	11.22
B5	0	0	0	0	0	0	0	0
B6	2	3	1	22.44	7	7	3	63.58
B7	0	0	0	0	1	1	0	7.48
B8	0	1	0	3.74	0	1	0	3.74
B9	0	0	0	0	1	0	0	3.74
B10	2	3	0	18.7	2	2	1	18.7
B11	0	0	0	0	0	0	0	0
Average				4.49				6.73

Table 22: SSQ results for translation participants

User ID	Pre				Post			
	Naus	OM	Dis	SSQ-score	Naus	OM	Dis	SSQ-score
C1	3	5	1	33.66	3	6	1	37.4
C2	4	3	2	33.66	3	5	3	41.14
C3	0	0	0	0	0	0	0	0
C4	0	0	0	0	4	0	1	18.7
C5	0	1	0	3.74	0	0	0	0
C6	1	3	1	18.7	2	3	1	22.44
C7	0	0	0	0	0	0	0	0
C8	0	0	0	0	1	0	1	7.48
C9	0	0	0	0	1	0	3	14.96
C10	2	8	3	48.62	3	4	4	41.14
C11	0	0	0	0	0	0	0	0
C12	0	0	0	0	2	1	1	14.96
C13	0	0	0	0	2	0	0	7.48
C14	0	0	0	0	1	0	0	3.74
Average				9.88				14.96

Table 23: SSQ results for curvature participants

UserID	Height	Gait
A01	195.903	52.538
A02	179.806	60.881
A03	164.974	77.316
A04	176.980	76.297
A05	176.499	75.865
A06	168.533	-
A07	166.453	58.652
A08	167.972	25.485
A09	160.727	-
A10	164.094	63.390
A11	146.933	53.201
A12	148.986	-
A13	157.082	-
A14	163.827	48.283
A15	179.389	73.232
A16	165.142	49.284
A17	180.946	49.667

Table 24: Eye height and gait of rotation gain participants

UserID	Height	Gait
B01	176.706	72.627
B02	163.449	78.725
B03	191.715	51.221
B04	165.039	-
B05	168.923	48.841
B06	168.127	-
B07	177.756	52.784
B08	189.455	57.046
B09	178.051	69.509
B10	175.529	81.834
B11	153.639	-

Table 25: Eye height and gait of translation gain participants

UserID	Height	Gait
C01	175.766	45.064
C02	165.755	-
C03	154.694	61.244
C04	158.852	54.243
C05	167.242	70.428
C06	180.679	53.109
C07	175.779	70.459
C08	181.818	55.265
C09	161.586	63.566
C10	164.118	68.124
C11	174.433	-
C12	166.058	53.959
C13	131.672	90.361
C14	171.630	62.034

Table 26: Eye height and gait of curvature gain participants